Universidade de Lisboa

Faculdade de Ciências de Lisboa

Departamento de Estatística e Investigação Operacional


# Planning the distribution of agricultural products in a short distribution channel

Bruno Miguel Craveiro de Oliveira

Dissertação

Mestrado em Estatística e Investigação Operacional

Especialização em Investigação Operacional

Professora Maria da Conceição da Fonseca

Professora Isabel Maria de Jesus Martins

2013

## Resumo

Os custos logísticos são um fator importante no que respeita à competitividade das empresas em economia de mercado. Estes custos resultam das diferentes fases dos sistemas logísticos, nomeadamente a produção, o armazenamento, a distribuição, a gestão de informação, os serviços ao cliente, entre outros. As empresas podem assim melhorar a sua competitividade através da redução destes custos e, portanto, do custo total associado aos bens e serviços providênciados.

Este trabalho tem por base uma iniciativa transnacional promovida em Portugal pela ADREPES (Associação para o Desenvolvimento Regional da Península de Setúbal), em parceria com a congénere da região francesa *Pays du Mans*, designada por "Da Quinta para o Prato". Esta iniciativa pretende estabelecer um circuito curto de comercialização assente na criação e articulação de uma rede entre produtores e responsáveis de estabelecimentos de restauração públicos e privados da região de Setúbal. Atualmente, um conjunto de cinco agricultores e um conjunto de sete estabelecimentos estão interessados em participar na iniciativa. Os agricultores serão os responsáveis pelo planeamento da distribuição dos produtos. Este trabalho pretende apoiar os agricultores nesta tarefa.

A afetação da produção à procura, a determinação das rotas de distribuição dos produtos e em certa medida a gestão da armazenagem dos produtos armazenáveis são os problemas principais que se identificaram no planeamento da distribuição dos produtos. Neste trabalho, são apresentadas duas abordagens distintas para a resolução dos dois primeiros problemas. Numa abordagem integrada, a afetação e a determinação das rotas faz-se através de um único modelo de programação linear inteira mista e na outra abordagem, estes problemas são estudados separadamente, sendo resolvidos através de heuristicas. Estas abordagens foram incorporadas num sistema de apoio à decisão.

Este sistema, desenvolvido apenas com software gratuito, permite que os agricultores acedam a uma base de dados com informação sobre eles próprios, os clientes, os produtos, a produção, os stocks e a procura. Quer os agricultores quer os clientes podem submeter informação *on-line* sobre as produções e procuras, respetivamente. Com base na informação armazenada, o sistema estabelece semanalmente o plano diário da distribuição dos produtos entre os agricultores, os clientes, e o armazém no caso dos produtos armazenáveis.



**Palavras chave:** *Cadeias de abastecimento sustentáveis, Logística, Distribuição, Problemas de recolha e entrega, Programação linear inteira mista, Heurísticas.*

# Abstract

Logistics costs are an important factor in the competitiveness of firms in a market economy. These costs result from the different stages of logistics systems, such as production, storage, distribution, information management, customer services, among others. Firms may improve their competitiveness by reducing these costs, and thus the total cost of providing goods and services.

This work focuses on a transnational initiative promoted by ADREPES (Association for the Regional Development of Peninsula of Setúbal), in association with its French counterpart from *Pays du Mans*, entitled "Da Quinta para o Prato" ("From Farm to Plate"). The goal of this initiative is to establish a short circuit commerce network between farmers and public or private catering establishments in the region of Setúbal. At the time, a set of five farmers and a set of seven establishments are interested in the initiative. Farmers will be responsible for the distribution planning of their products. This work aims to provide a decision support system to help the farmers in their task.

Assignment of production to demand, determination of the distribution routes and in a certain way stock management of storable products were the main problems identified in the planning of the distribution of the production. In this work, two distinct approaches for the first two problems are presented. In the approach that one calls integrated, assignment and routes determination are solved by a single mixed integer linear programming model, and in the other approach, these problems are addressed separately and solved by heuristics. These approaches were introduced in a decision support system.

This system, developed with only free software, allows farmers to have access to a database with information about themselves, customers, products, production, stock and demand. Both farmers and customers can submit information on-line about productions and demands, respectively. Based on stored data, the system establishes weekly the daily plan of products distribution between farmers, customers, and the warehouse for the storable products.



**Keywords:** *Sustainable supply chains, Logistics, Distribution, pickup and delivery problems, Mixed integer linear programing, Heuristics.*

# Acknowledgments

This work represents the end of a long academic journey where I was fortunate enough to meet and work with remarkable people, colleagues and professors alike, which in way or another helped me to achieve this goal.

I deeply thank my mother and sister for all the support throughout this endeavor. Without them I would have never come this far.

I would also like to thank Professor Maria Conceição and Professor Isabel Martins for all the support, patience and guidance throughout the development of this work and Ricardo Magalhães for providing the initial concept of the presented software.

This work is dedicated to the loving memory of my father, whose spirit will always be with me.

# Contents

# List of Figures

# List of Tables

# List of Algorithms

# Chapter 1

# Introduction

The *Association for the Regional Development of Peninsula of Setúbal* (ADREPES), in association with its French counterpart from *Pays du Mans*, is developing a transnational project, named "Da Quinta Para o Prato" ("Farm to Plate"), which is set to establish a local food system in the Setúbal region.

It is difficult to find a unique definition for a *local food system*, as different aspects may be taken into account, but there are common characteristics that can be summarized as follows: existence of collaborative and integrated work, focus on local production and distribution, integration of the values of equity and social justice and a concern for environmental issues. "A *community food system* is one in which sustainable food production, processing, distribution and consumption are integrated to enhance environmental, economic, social and nutritional health of a particular place" [6]. For a discussion on local food systems based on a case study in Sweden we refer to [2] and [3].

Short distribution channels can be classified into direct-to-consumer and direct-to retail. In a *direct-to-consumer* channel of agricultural products, farmers sell their products directly to the final consumer. In a *direct-to-retail* channel, there is at most one intermediary (for example, when products are sold to schools, hospitals, canteens, restaurants and other organizations) between the farmer and the final consumer. At the time, Da Quinta para o Prato is only interested in developing a short distribution channel, where farmers from the region of Setúbal would supply their productions mainly to local canteens and restaurants.

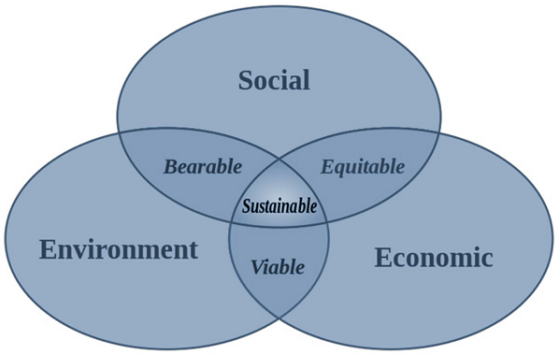



Figure 1.1: Sustainability pillars.

In order to comply with the underlying social and environmental values of the overall project's devised

food system, it is important that the initiative's operations follow more than a "profit first" approach. The sustainability of an initiative can be analyzed by what is called a Triple Bottom Line (TBL) approach (Figure 1.1), which takes into account three main criteria, *social responsibility*, *environmental stewardship* and *economic viability*, sometimes succinctly described as "People, Planet, Profit", as first coined by John Elkington in 1995.

Concerning social responsibility, some ordering criteria are used in this work to provide a fair division of benefits amongst all stakeholders. Regarding environmental stewardship, the initiative's main activity, which is to produce local agricultural products, has in itself positive environmental externalities, such as maintaining a well cared agricultural landscape in the periphery of the urban area of Setúbal. Besides, the establishment of such a short distribution channel will help to reduce the overall distance traveled by food, from the production sites to the final consumer, commonly referred to as *food miles*, contributing in this way for the reduction of greenhouse gas emissions. Regarding economic viability, it is necessary to ensure that the initiative's provision of goods remains profitable.

The initiative's economic activity can be divided into three different stages or processes: product production, storage and distribution. Note that most of the supply chain is vertically integrated, this is, the initiative controls most of these processes. This work focus solely on storage and distribution stages, optimizing the weekly distribution plans as to obtain a balanced participation of farmers and a balanced satisfaction of customers, as well as less costly distribution routes. Transportation costs represent a relevant component in a product's final price (generally from 10% to 20%), and therefore the main aim is to minimize transportation costs, which will hopefully contribute for the initiative's economic viability.

## 1.1 Motivation

The successful establishment of such a short distribution channel would provide several social, economic and environmental benefits. It would allow farmers to have access to new markets and new sources of income, selling products that currently are not sold, and that otherwise would be wasted. Customers, on the other hand, would have access to local products of quality, hopefully at competitive prices, delivered at their own places. The community would benefit from the quality of these products and also from a greater sense of trust between customers and farmers, contributing thus for the initiative's social responsibility and economic viability. Environmental benefits would result essentially from short distances traveled by food and the maintenance of the rural landscape in the periphery of the urban area. As transportation costs can be a significant part of a business overall costs, they can make products more expensive and therefore less competitive. It is then advisable to plan the food distribution in a rational manner in order to contribute for the initiative's economic viability.

## 1.2 Objectives

This work presents the current stage of the development of a *Decision Support Software* (DSS), which will allow farmers to collect, store and modify all relevant data and to establish weekly distribution plans. This DSS can be decomposed in three major sections: data collection, data storage and optimization procedures applied to the distribution planning problem. In order to collect the necessary data, a website was developed as a user-friendly platform, where new customers and farmers can sign in and all customers and farmers can submit their weekly product requests and production yields, respectively. In order to store all collected data, a database was developed and connected to the website.

The distribution planning problem encompasses the assignment of production to demand and the determination of the distribution routes. The assignment of production to demand takes into account that some products are storable and thus there is, in a certain way, stock management. One considered two types of distribution routes: pickup and delivery routes, in which products are transported from farmers or from the warehouse to customers and, whenever there is some available capacity in the vehicles, storable products are also transported from farmers to the warehouse; pickup routes in which storable products are brought from farmers to the warehouse. Concerning the optimization methods, two different approaches were considered in order to address the distribution planning problem. First, a mixed integer linear programming (MILP) model is proposed where both problems, the assignment of production to customers' demand and the determination of the pickup and delivery routes, are solved simultaneously as an "integrated" problem. In the "unaggregated" approach, those problems are addressed separately and they are both solved using heuristics. Stock management, the assignment of production to the warehouse and the determination of the pickup routes for storable products only are solved separately.

## 1.3 Document Structure

This dissertation is composed of eight chapters including this one. In the next chapter, one presents the characteristics of the case study and the basic notation that is used throughout the text. In section 3, one presents an overview of the developed stages of the DSS: from data collection and data storage to stock management and the actual main procedure which determines the weekly distribution plans. In chapter 4, we present a MILP model for the integrated problem. Chapter 5 is dedicated to the unaggregated approach, where two heuristics to solve the problems of the assignment of production to customers' demand and pickup and delivery routes determination are presented. For the assignment problem, a heuristic assigns production to customers' demand, taking into account some equity criteria. For the determination of the pickup and delivery routes, one proposes a two phase heuristic composed of a constructive phase, based on the savings concept presented by Clarke and Wright, and a combined improvement phase, where two different local procedures, each one guided by a simulated annealing procedure, are executed in turns. Chapter 6 is dedicated to the assignment of production to the warehouse and the determination of the pickup routes for

storable products only. In chapter 7, computational results for both optimization approaches, obtained for some instances based on the case study are presented. In the last chapter, one presents the final conclusions and future work.

# Chapter 2

# Case Study and Basic Notation

The presented case study addresses the activity of a group of farmers in the area of Setúbal which intend to directly sell their production to local restaurants and canteens. It was established that distribution plans should be devised on a weekly basis.

At the time, the initiative is composed by five farmers, which are located in the rural area of the Setúbal region. They produce fresh and storable products. It is desirable that fresh products are picked up and delivered in the same day. For the storable products, there is information about their production and storage periods. For the computational tests, farmers provided information about their daily productions for the month of March of 2012. For this month, farmers produced amongst themselves eighteen different products, fifteen fresh products and three storable products (potatoes, onions and carrots). There are seven restaurants and canteens interested in the initiative. These customers are somewhat clustered in the urban area of the Setúbal region.

Farmers are able to provide their monthly estimated production yields, based on past years' information, and daily production yields on a weekly basis. Customers are able to plan the weekly demand and also provide the monthly estimated demand of storable products based on past years' information. Farmers and customers have preferences on the period of the day in which products should be picked up and delivered, respectively. There are customers who wish not to receive a product if their requests for this product cannot be fully satisfied.

There is a depot, where vehicles are kept, and a warehouse to store products, for which no maximum capacity was mentioned. Both, depot and warehouse, are located at the same geographical location, which coincides with one of the farmers. At the time, two vehicles are available with a given capacity of 400 Kg each.

Next, one presents some of the notation used to mathematically describe the case study and other instances. This notation is used throughout the text. The remaining notation is introduced when needed.

- $P$ - set of products

- $Ps$ - set of storable products
- $Pf$ - set of fresh products
- $Cl$ - set of customers
- $Cl_{tot}$ - set of customers that prefer not to receive a product if their product request is not entirely satisfied
- $Fa$ - set of farmers
- $AR$ - the warehouse
- $q_{ikd}$ - demand of product $i \in P$ of customer $k \in Cl$ in day $d$
- $p_{ijd}$ - amount of product $i \in P$ available in farmer $j \in Fa$ or in the warehouse $j = AR$ in day $d$
- $pa_{id}$ - stock of product $i \in P$ at the beginning of day $d$
- $e_{ijkd}$ - quantity of product $i \in P$ delivered to customer $k \in Cl$ or to the warehouse $k = AR$ in day $d$, from farmer $j \in Fa$ or from the warehouse $j = AR$ ($e_{i,AR,AR,d}$ is not defined).

# Chapter 3

# Development of the Decision Support System

## 3.1 Data Collection

All data regarding products consumption and production was collected through questionnaires. In an attempt to provide customers and farmers, for each week, with a convenient way of submitting daily requests and productions, respectively, a website was developed (Figures 3.1, 3.2, 3.3, 3.4 and 3.5). This platform also allows the user to register as an active member of the project. The website was developed using *HTML* with embedded *PHP* [18] and it is currently located at a Portuguese web-host server and its URL is www.projectodaquinta.com/.

One aspect of data collection worth mentioning is that all distances and travel times were obtained in a non-automatic way. These parameters were obtained through the Michelin's website *www.viamichelin.pt*, which somehow gives the shortest and quickest route between two locations. This proved to be a rather inefficient way of obtaining data. The use of a *geographical information system* (GIS) tool embedded in the DSS, which would automatically calculate distances and travel times, would be an efficient alternative.

## 3.2 Data Storage

In order to store all initiative's data, a local database was created using *MySQL* [11]. This database holds information about customers and farmers which are members of the project. It also has products' information, daily requests and production yields on a weekly basis and deliveries (see Appendix A). In order to store all data submitted through the website, a database was also created in a remote server, using *phpMyAdmin*, a free software tool written in *PHP*, intended to handle the administration of *MySQL* over the web. This software was provided by the website host. Note that in order to access data faster, all data was imported


Figure 3.1: Website home page.


Figure 3.2: Customers' enrollment form.


Figure 3.3: Farmers' enrollment form.

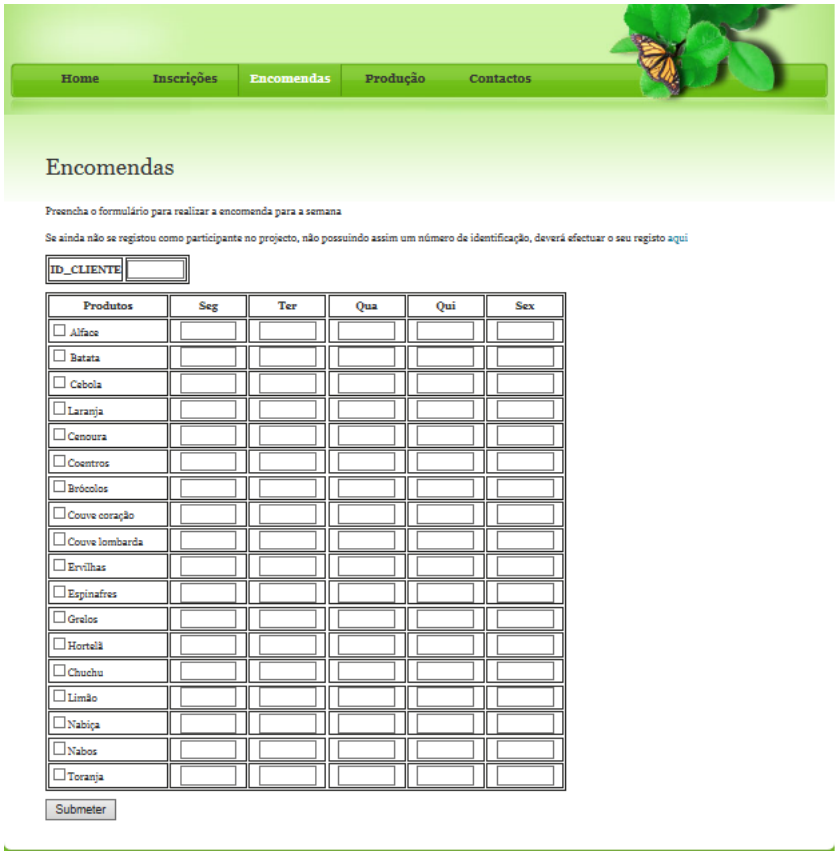



Figure 3.4: Weekly requests form.

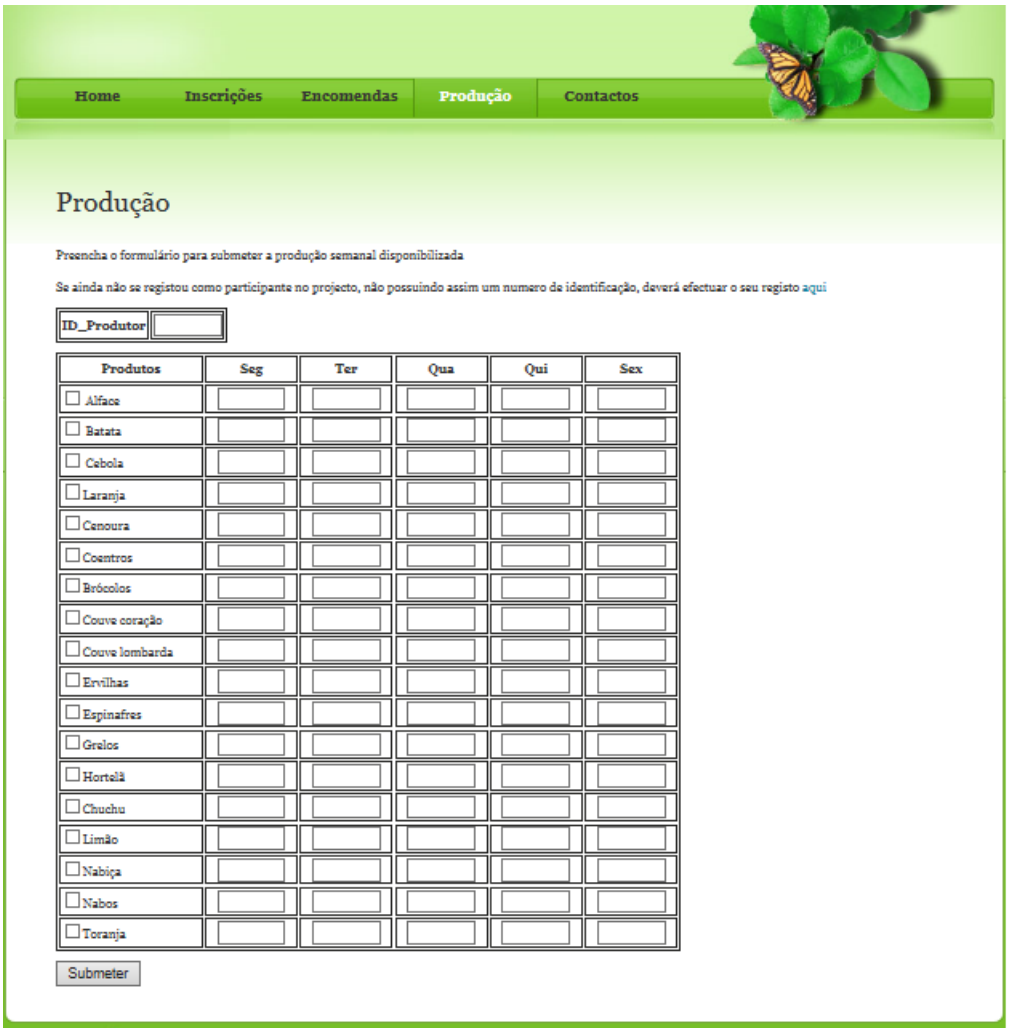



Figure 3.5: Weekly productions yields form.

from the remote database to the local database.

## 3.3 Software Overview

In order to provide farmers with a tool with which they could essentially a) access all relevant data of the initiative, b) assign production to demand and c) determine distribution routes, a *graphic user interface* (GUI) based software started to be developed using *JAVA* [13].

At the time, this software gives the administrator access to general information about customers, farmers and products, as well as weekly product requests and productions yields (Figures 3.6, 3.7, 3.8). It also allows the administrator to determine the architecture of the weekly distribution plan. In fact, the administrator can establish which days of the week are delivery days and whether a specific problem related to storable products - the determination of pickup routes for collecting storable products only - is solved or not.


(a) Main display.


(b) Customers' information.

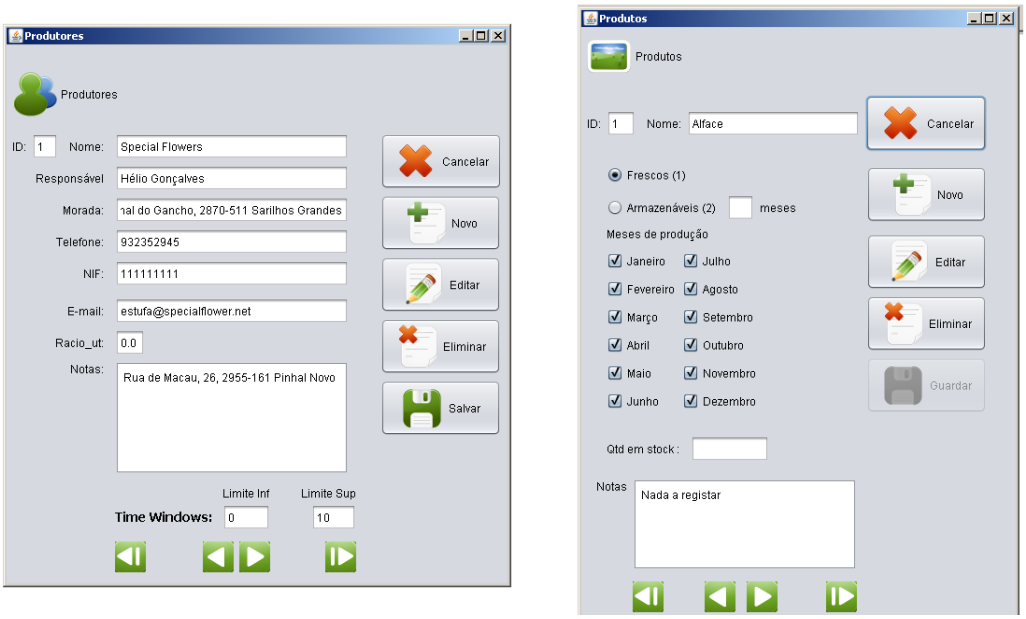
Figure 3.6: Software display - Main display and customers' information.

The weekly distribution plan is defined at the beginning of the week (Algorithm 1 displayed in Figure 3.9). The quantities of storable products to be picked up over the week taking into account current and future demands are determined. Part of these quantities are brought to the warehouse. This stock management attempts to prevent shortage of supply essentially during non-productive periods. For every delivery day, production is assigned to customer's demands and distribution pickup and delivery routes are established. In these routes, products are picked up at the farmers or loaded at the warehouse and delivered to customers. At the same time, storable products are also brought to the warehouse if vehicles' capacities are not exceeded. Finally, at the end of the week, it is possible to establish pickup routes for collecting storable products, according to the quantities that are still necessary to bring to the warehouse. The administrator can decide if these quantities are sufficient to justify the collection ("if criterion is met" in Algorithm 1).

To contribute for the initiative's economic viability and environmental stewardship through the reduction of food miles, the main objective is then to minimize the total route distance (or some measure of cost) traveled by the vehicles that transport the products. To act upon social responsibility, two other goals are kept in mind, a fair participation of farmers and a balanced satisfaction among customers.

## 3.4 Stock Management

Farmers do not have greenhouses and thus the majority of the products produced by them are seasonal. Some of these products are storable (potatoes, onions and carrots). For these products, it is desirable to maintain appropriate levels of stock in order to satisfy demands during the longest periods of time. To establish the quantities of storable products to be picked up during each week, taking into account this goal, a simple provision planning procedure was implemented (see Algorithm 2).




(a) Farmers' information.

(b) Products' information.

Figure 3.7: Software display - Farmers' and products' information.

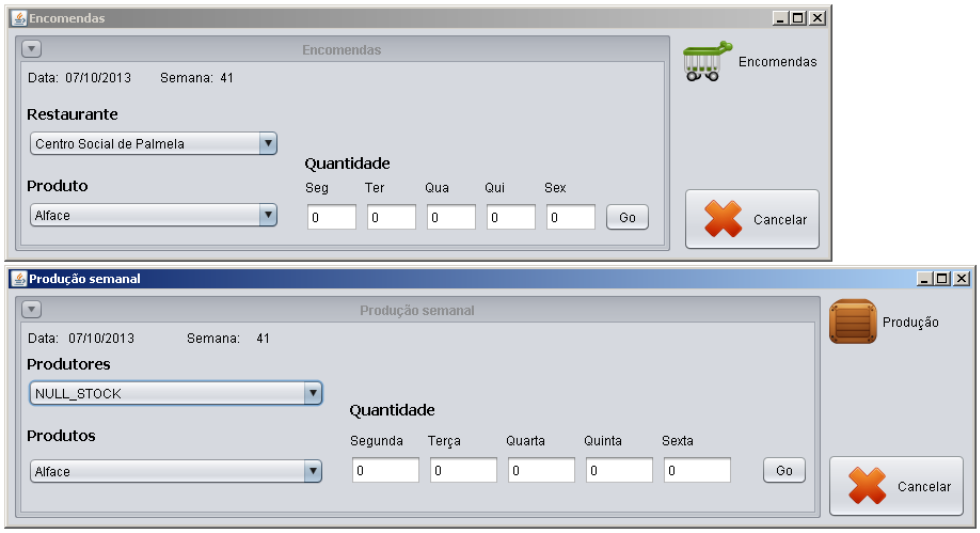



Figure 3.8: Software display - Weekly requests and production yields.

Algorithm 2 is only executed in the first day of the Weekly procedure. For each storable product, the procedure uses the estimated monthly demand previously provided by customers to calculate, roughly, the quantity that is necessary to pick up in each week of production, in order to satisfy the demand in that week and in the following storage period. These quantities are the same for all weeks of production and can be perceived as rectified weekly demands.

The procedure also uses the estimated monthly production made available by farmers, converting it into

**Algorithm 1** Weekly procedure

```
for every day d of the week do
    if d = Monday then
        Stock management
    end if
    if d is delivery day then
        Assignment of production to demand
        Distribution routes planning
    end if
    if d = Friday then
        if criterion is met then
            Final distribution routes planning
        end if
    end if
end for
```

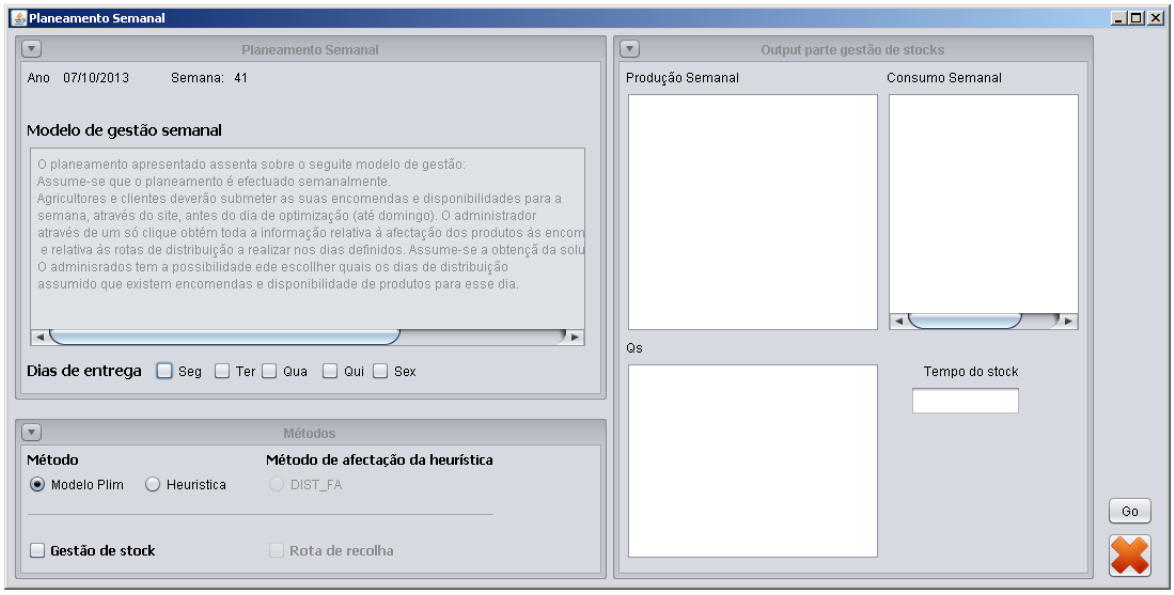



Figure 3.9: Weekly procedure.

average weekly values. For each week $w$ and for each storable product $i$ in production, the procedure proceeds as follows:

- marks the weeks of the following storage period in which the corresponding week's production is insufficient to fully satisfy that week's rectified demand;
- assigns production of $w$ to $w$ according to the rectified demand and if there is a production surplus,

the remainder production is successively assigned to the previously marked weeks in order to satisfy their rectified demand.

In order to execute the procedure the following data is necessary:

- $\hat{Tp}_i$ - production period of product $i \in P_s$, in months (consecutive productive months)
- $Tp_i$ - production period of product $i \in P_s$, in weeks (consecutive productive weeks)
- $\hat{Ta}_i$ - storage period of product $i \in P_s$ that follows period $Tp_i$, in months
- $\hat{pr}_{iw}$ - expected production of product $i \in P_s$ in week $w \in Tp_i$
- $\bar{C}_i = \frac{\sum_{m \in \hat{Tp}_i \cup \hat{Ta}_i} C_{im}}{|\hat{Tp}_i| 4.36}$ - needed quantities of product $i \in P_s$ in each week of production, in order to satisfy the demand in this week and in the following storage period, where $C_{im}$ is the expected demand of product $i$ in month $m \in \hat{Tp}_i \cup \hat{Ta}_i$, and 4.36 is the average number of weeks in a month.

The reason for the simplistic calculation of $\bar{C}_i$ is related to the fact that, in the case study, the monthly demand of each storable product is practically constant over the year.

For a given week $W$:

- $D_W$ - set of delivery days that comprise week $W$
- $Cl_{iW}$ - set of customers that ordered product $i \in P_s$ in week $W$, $Cl_{iW} = \{k \in Cl : \sum_{d \in D_W} q_{ikd} > 0\}$.

For every product $i \in P_s$ such that $W \in Tp_i$:

- $Ta_{iW}$ - storage period following week $W$, in weeks
- $pr_{iW}$ - production in week $W$, $pr_{iW} = \sum_{j \in Fa} \sum_{d \in D_W} p_{ijd}$
- $pa_{iW}$ - stock in the beginning of week $W$
- $q_{iW}$ - demand in week $W$, $q_{iW} = \sum_{k \in Cl_{iW}} \sum_{d \in D_W} q_{ikd}$.

The output of the procedure is the following:

- $[Q_{iW}]$ - matrix, where $Q_{iW}$ is the quantity to be picked up in week $W$ of product $i \in P_s : W \in Tp^i$

At this time, addressing this problem as a typical problem of stock management [15] is beyond the scope of this work, since priority has been given to assignment of production to demand and the determination of the pickup and delivery routes.

**Algorithm 2** Provision planning procedure, for a given week $W$

**for all** $i \in Ps : W \in Tp^i$ **do**
  $pra_i \leftarrow pr_{iW} + pa_{iW}$
  $f \leftarrow \max\{\bar{C}_i, q_{iW}\}$
  **if** $pra_i \leq f$ **then**
    $Q_{iW} \leftarrow pra_i$
  **else**
    $B = \{\text{weeks } w \in Ta_{iW} : \hat{pr}_{iw} < \bar{C}_i\}$
    $d \leftarrow pra_i - f$
    $Q_{iW} \leftarrow f$
    **while** $d > 0 \wedge B \neq \emptyset$ **do**
      **for all** $w \in B$ **do**
        $e \leftarrow \min\{d, \bar{C}_i - \hat{pr}_{iw}\}$
        $Q_{iW} \leftarrow Q_{iW} + e$
        $d \leftarrow d - e$
        $B = B \setminus \{w\}$
      **end for**
    **end while**
  **end if**
**end for**
**return** $[Q_{iW}]$

## 3.5 Satisfaction Ratio for Customers

If in a given day, the production of a product(plus stock in case of a storable product) is insufficient to fully satisfy the demand then, corrections are made to customers' requests, in order to guarantee feasibility, taking into account products' availability, customers' preferences (some prefer not to receive a product if it is not possible to completely supply that product's request) and customers' priority given by a satisfaction ratio. Sorting customers in a non-descending order according to this satisfaction ratio assures that less satisfied customers will see their requests attended first, therefore contributing for a fairest distribution of products.

First some notation needs to be introduced before the definition of the satisfaction ratio. For a given day $d \in D_W$:

- $Cle_d$ - set of customers who placed an order at day $d \in D_W$, $Cle_d = \{k \in Cl: \sum_{i \in P} q_{ikd} > 0\}$.

For every customer $k \in Cle_d$:

- $q_{ik}$ - demand of product $i \in P$ until the delivery day before $d$, $q_{ik} = \sum_{t=1}^{d-1} q_{ikt}$

- $e_{ik}$ - quantity effectively delivered of product $i \in P$ until the delivery day before $d$, $e_{ik} = \sum_{t=1}^{d-1} \sum_{j \in Fa \cup \{AR\}} e_{ijkt}$
- $hc_{ik}$ - fraction of the demand of product $i \in \bar{P}c_k$, where $\bar{P}c_k = \{i \in P : q_{ik} > 0\}$, effectively delivered until the delivery day before $d$, $hc_{ik} = \frac{e_{ik}}{q_{ik}}$.

*Satisfaction ratio* is

- $Hc_k = \frac{\sum_{i \in \bar{P}c_k} hc_{ik}}{n}$, where $n = |\bar{P}c_k|$.

As historical data needs to be erased from time to time due to limitations in data storage capacity, the satisfaction ratio is defined as a customers' attribute which is updated day by day, and is calculated, if $q_{ikd} > 0$, as $Hc_k \leftarrow 0.5 \times Hc_k + 0.5 \times \sum_{i \in P} \frac{\sum_{j \in Fa \cup \{AR\}} e_{ijkd}}{q_{ikd}}$, where for the very beginning $Hc_k \leftarrow 0$. $Hc_k$ values are between zero and one. Note that the new satisfaction ratio is a linear combination of two ratios, the historical and the daily ratios. It is established that both ratios have the same importance, but different linear combinations can be considered.

Customers are then ordered by non-descending order accordingly to their satisfaction ratio resulting in the $Cle_d^{ord}$ vector of ordered customers.

## 3.6 Participation Ratio for Farmers

To guarantee that all farmers deliver products, a participation ratio is calculated. This calculation is only noticeable when more than one farmer produces the same product, this is, there is competition between farmers. Participation ratio is simply the ratio between the whole production actually delivered over the whole production. Note that, if customers mind paying different prices for the same product (it is assumed that the quality of a product does not depend on its origin), then the price of a product needs to be the same regardless whom might have produced it. One may take into account that production costs may differ from farmer to farmer due to different production techniques or presence of economies of scale. To take this aspect into consideration, or in other words to establish a fair split of gains, the participation ratio can also be defined as the ratio between the profit actually obtained over the profit that would be obtained if the whole production was delivered.

First, some notation need to be introduced.

For a given delivery day $d \in W_D$:

- $Fap_d$ - set of farmers with production, $Fap_d = \{j \in Fa : \sum_{i \in P} p_{ijd} > 0\}$.

For each farmer $j \in Fap_d$:

- $p_{ij}$ - production of product $i \in P$ until the delivery day before $d$, $p_{ij} = \sum_{t=1}^{d-1} p_{ijt}$

- $ef_{ij}$ - production of product $i \in P$ effectively delivered until the delivery day before $d$, $ef_{ij} = \sum_{t=1}^{d-1} \sum_{k \in Cl \cup \{AR\}} e_{ijkt}$
- $hf_{ij}$ - fraction of production of product $i \in \bar{P}f_j$, , where $\bar{P}f_j = \{i \in P : p_{ij} > 0\}$, effectively delivered until the delivery day before $d$, $hf_{ij} = \frac{ef_{ij}}{p_{ij}}$.

*Participation ratio* is

- $Hf_j = \frac{\sum_{i \in \bar{P}f_j} hf_{ij}}{n}$, where $n = |\bar{P}f_j|$.

As previously referred to for the satisfaction ratio for customers, historical data needs to be deleted from time to time due to current data storage capacity limitations. Participation ratio is defined as a farmers' attribute which is updated day by day, and is calculated, if $p_{ijd} > 0$, as $H_j \leftarrow 0.5 \times H_j + 0.5 \times \sum_{i \in P} \frac{\sum_{k \in Cl \cup \{AR\}} e_{ijkd}}{p_{ikd}}$. $Hf_j$ values are between zero and one. It was established that the daily participation ratio is as important as the historical ratio, but different linear combinations of both can be considered.

# Chapter 4

# Integrated Approach

For each delivery day of the week, it is necessary to properly assign production (and stock for storable products) to customers' demand and to determine the pickup and delivery routes that minimize the total transportation cost. These problems can be considered simultaneously, giving what one calls the integrated approach.

If in a given day, production (and stock for storable products) can not satisfy all demand, then a correction procedure is applied.

After making the necessary requests' corrections, an integrated MILP model is then used, not only to determine which farmers or if the warehouse will supply each customer with which products and quantities, but also to establish the pickup and delivery routes of a fleet of vehicles that will minimize the transportation costs.

## 4.1 Demand Adjustment

For any given day $d \in D_W$, the demand adjustment procedure (Algorithm 3) is applied. The output of this procedure is a three-dimensional matrix $[e_{ikd}]$, where $e_{ikd}$ is the quantity of product $i \in P$ delivered to customer $k$ in day $d$. This procedure assigns production to demand in an orderly fashion, taking into account customers' satisfaction ratio. This will guarantee feasibility while pursuing a balanced distribution of products among the customers.

## 4.2 MILP Model

The proposed model is based on the general pickup and delivery model with time windows presented by Salvelsberg and Sol (1995) [14]. Unlike the general model, where requests origins are known in advance, the proposed model does not establish beforehand which origins (farmers) will supply which customers, leaving that decision to the model. The demand must be satisfied, the supply does not exceed the offer and, for

**Algorithm 3** Demand adjustment procedure, for a given day $d$

**for all** $i \in P$ **do**
  $p_{id} \leftarrow \sum_{j \in Fa} p_{ijd}$ if $i \in Pf$
  $p_{id} \leftarrow \sum_{j \in Fa} p_{ijd} + pa_{id}$ if $i \in Ps$
**end for**
**for all** $k \in Cle_d^{ord}$ **do**
  **for all** $i \in P : q_{ikd} > 0 \wedge p_{id} > 0$ **do**
    **if** $k \in Cl_{tot}$ **then**
      **if** $q_{ikd} \leq p_{id}$ **then**
        $e_{ikd} \leftarrow q_{ikd}$
        $p_{id} \leftarrow p_{id} - e_{ikd}$
      **else**
        $e_{ikd} \leftarrow 0$
      **end if**
    **else**
      $e_{ikd} \leftarrow \min\{q_{ikd}, p_{id}\}$
      $p_{id} \leftarrow p_{id} - e_{ikd}$
    **end if**
  **end for**
**end for**
**return** $[e_{ikd}]$

the supply of storable products, the warehouse is chosen first. The model also takes into account that each customer requests several products and that each of these requests can be satisfied by more than one farmer. These requests must be delivered all at the same time (a customer is visited only once), therefore, all customer's suppliers must be visited before the customer. Both, customers and farmers, have time intervals in which they can be visited. The capacity of each vehicle cannot be exceeded and all vehicles begin and end their route in the depot. The fleet is considered to be heterogeneous.

This model is thus a combination of an assignment problem and a pickup and delivery problem, where the objective is to minimize the transportation costs. Pickup and delivery problems discussed in the literature have different characteristics as one can see, for example, in the survey articles [8], [9] and [10], and in the secondary literature given there or in other as [4] and [7]. Anyway, the pickup and delivery problem in this work can be perceived as a static single-depot asymmetric multi-commodity pickup and delivery problem with time windows. The problem can be defined as static since all information is known in advance. A single depot is used and the asymmetry arises from the use of distances given by the present road network. The problem is multi-commodity since customers and farmers can request and supply more than one product.

The mathematical model proposed is described as follows. Let $G = (V, E)$ be a graph where $V$ is the node set partitioned into three subsets: $O$, the set of initial and ending nodes for the vehicles, $O^+$ and $O^-$ respectively, $N^-$, the set of customers, and $N^+$, the warehouse and farmers' replicas. $O^+$ and $O^-$ correspond to the single depot of the vehicles.

Replicas allow farmers to be visited more than once in the same route if necessary. In fact, the same vehicle may return to a visited farmer (another replica of the farmer) to pick up products for another customer (see Figure 4.1). For each customer, there is a replica of each farmer and the warehouse. Therefore, $|N^+| = |\mathcal{C}l| \times |(AR \cup \mathcal{F}a)|$. All distances between replicas of the same farmer are set to zero. The distances between two replicas of the same farmer and any other location are equal, as well as the distances in the opposite direction. Service activities other than product loading are essentially performed as the vehicle reaches the farmer, such as, verifying quantities to be loaded and papers related to the transport and addressing inconformities. So, service times are also considered to be null between replicas of the same farmer. Non-null service times are considered to be constant, that is, do not depend on the quantities to be loaded or unloaded or the number of customers to be supplied. Each node $i \in V$ is associated with a time interval $[a_i, b_i]$ (time window), where products can be picked up and delivered.

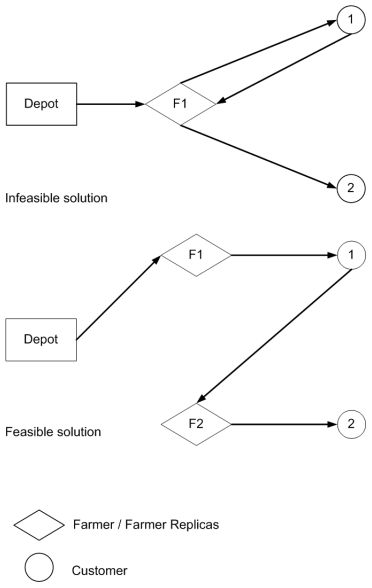



Figure 4.1: Replicas example.

In order to reduce the size of the graph, arc set $E$ is the result of a preprocessing phase where problem characteristics are taken into account. The construction of the arc set $E$ is described next. First, the following notation is introduced.

- $M$ - set of vehicles
- $Q_v$ - capacity in weight of vehicle $v \in M$.

For a given day $d \in D_W$:

- $e_k$ - weight of the whole demand of customer $k \in N^-$ in day $d$, $e_k = \sum_{i \in P} e_{ikd}$.

**Graph construction** Let $L = N^+ \cup N^-$. The complete graph between replicas and customers would be given by the set of arcs $L^2 = \{(i, j) : i, j \in L\}$. Let $E = L^2$.

- *Farmers and the warehouse.* Vehicles leave the depot empty, and therefore, the first location to be visited by a vehicle can only be a replica. It is then established that from the start location ($O^+$) there are only directed arcs to nodes $j \in N^+$; $E \leftarrow E \cup \{(O^+, j) : j \in N^+\}$.

- *Customers.* No products are brought to the depot, and therefore, there are only directed arcs to the final location ($O^-$) coming from customers $k \in N^-$; $E \leftarrow E \cup \{(k, O^-) : k \in N^-\}$.
- *Precedences.* Due to precedence constrains, all directed arcs $(k, j)$ with $k \in N^-$ and $j \in N_k^+$ are removed from $E$; $E \leftarrow E \setminus \{(k, j) : k \in N^-, j \in N_k^+\}$.
- *Vehicle capacity.* If $e_k + e_{k'} > Q_v$ where $k, k' \in N^-$, then arcs $(k, k')$ are removed from $E$; $E \leftarrow E \setminus \{(k, k') : e_k + e_{k'} > Q_v,\ k, k' \in N^-\}$.
- *Self-loops.* There are no self-loops (arcs from a node to itself), and therefore, arcs $(l, l)$ where $l \in L$ are removed from $E$, $E \leftarrow E \setminus \{(l, l) : l \in L\}$.
- *Vehicle usage.* As the number of vehicles is established upfront but it might not be necessary to use them all, a directed arc from the start location $O^+$ to the end location $O^-$ is added; this arc will be used if one vehicle $v \in M$ at least is not used; $E \leftarrow E \cup \{(O^+, O^-)\}$.

Next, the definition of a pickup and delivery route for a vehicle is presented.

**Definition 1** *A pickup and delivery route $R_v$ for a vehicle $v \in M$ is a directed route through a subset $V^v \subseteq V$ such that:*

1. *$R_v$ starts at the depot $O^+$*
2. *If a replica from $N_k^+$ is visited, it is visited before customer $k \in N^-$*
3. *Vehicle $v \in M$ visits each location in $V^v$ exactly once*
4. *A replica from $N_k^+$ is visited if it supplies customer $k \in Cl$*
5. *Vehicle's load never exceeds its capacity $Q_v$, $v \in M$*
6. *All locations are visited within their respective time windows*
7. *$R_v$ ends at the depot $O^-$.*

Before presenting the model, the following notation is added.

- $NF_j^+$ - set of replicas of farmer $j \in Fa$ or the warehouse $j = AR$
- $d_{ls}$ - distance between locations $l$ and $s$, $(l, s) \in E$
  - $d_{ls} = 0,\ j \in Fa,\ l, s \in NF_j^+$
  - $d_{l_1,s} = d_{l_2,s}$ and $d_{s,l_1} = d_{s,l_2},\ j \in Fa,\ l_1, l_2 \in NF_j^+,\ (l_1, s), (l_2, s), (s, l_1), (s, l_2) \in E$
  - $d_{O^+,O^-} = 0$
- $t_{ls}$ - travel time between locations $l$ and $s$, $(l, s) \in E$

- $t_{ls} = 0,\ j \in Fa,\ l, s \in NF_j^+$
- $t_{l_1,s} = t_{l_2,s}$ and $t_{s,l_1} = t_{s,l_2},\ j \in Fa,\ l_1, l_2 \in NF_j^+,\ (l_1, s), (l_2, s), (s, l_1), (s, l_2) \in E$
- $t_{O^+,O^-} = 0$

- $w_{ls}$ - service time at location $l \in V$, $(l, s) \in E$
  - $w_{ls} = 0,\ j \in Fa,\ l, s \in NF_j^+$

- $c_{ls}$ - travel cost between locations $l$ and $s$, $(l, s) \in E$
  - $c_{ls} = 0,\ j \in Fa,\ l, s \in NF_j^+$
  - $c_{l_1,s} = c_{l_2,s}$ and $c_{s,l_1} = c_{s,l_2},\ j \in Fa,\ l_1, l_2 \in NF_j^+,\ (l_1, s), (l_2, s), (s, l_1), (s, l_2) \in E$
  - $c_{O^+,O^-} = 0$

- $\underline{D}_l$ - minimum arriving time at location $l \in V \cup O$
- $\overline{D}_l$ - maximum arriving time at location $l \in V \cup O$.

The decision variables are the following:

- $z_k^v = \begin{cases} 1 & \text{if customer } k \in Cl \text{ is visited by vehicle } v \in M \\ 0 & \text{otherwise} \end{cases}$
- $g_l^v = \begin{cases} 1 & \text{if location } l \in N^+ \text{ is visited by vehicle } v \in M \\ 0 & \text{otherwise} \end{cases}$
- $x_{ls}^v = \begin{cases} 1 & \text{if vehicle } v \in M \text{ travels from location } l \text{ to location } s, (l, s) \in E \\ 0 & \text{otherwise} \end{cases}$
- $f_{il}$ - quantity of product $i \in P$ to be picked up from location $l \in N^+$
- $y_l$ - load (in weight) of a vehicle when it arrives at location $l \in V \cup O$
- $D_l$ - arriving time of a vehicle at location $l \in V$
- $D_{O^-}^v$ - arriving time of vehicle $v \in M$ at location $O^-$.

The model for a given day $d \in D_W$ is the following.

$$\min \sum_{v \in M} \sum_{(l,s) \in E} c_{ls} x_{ls}^v + \sum_{v \in M} \sum_{l \in N^+} g_l^v \tag{4.1}$$

$s.t$

$$\sum_{v \in M} z_k^v = 1 \qquad \forall k \in N^- \tag{4.2}$$

$$g_l^v \leq z_k^v \qquad \forall k \in N^-, l \in N_k^+, v \in M \tag{4.3}$$

$$\sum_{s:(l,s) \in E} x_{ls}^v = g_l^v \qquad \forall l \in N^+, v \in M \tag{4.4}$$

$$\sum_{s:(k,s) \in E} x_{ks}^v = z_k^v \qquad \forall k \in N^-, v \in M \tag{4.5}$$

$$\sum_{s:(l,s) \in E} x_{ls}^v - \sum_{s:(s,l) \in E} x_{sl}^v = 0 \qquad \forall l \in V, v \in M \tag{4.6}$$

$$\sum_{s \in N^+ \cup O^-} x_{O^+ s}^v = 1 \qquad \forall v \in M \tag{4.7}$$

$$\sum_{s \in N^- \cup O^+} x_{s O^-}^v = 1 \qquad \forall v \in M \tag{4.8}$$

$$f_{il} \leq \overline{U}^{il} (g_l^v + 1 - z_k^v) \qquad \forall i \in P, k \in N^-, l \in N_k^+, v \in M \tag{4.9}$$

$$\sum_{l \in N_{Arm}^+} f_{il} = \min\{p_{i,AR,d}; \sum_{k \in Cl} e_{ikd}\} \qquad \forall i \in P_s \tag{4.10}$$

$$\sum_{l \in NF_j^+} f_{il} \leq p_{ijd}; i \in P; j \in \mathcal{F}a \tag{4.11}$$

$$\sum_{l \in N_k^+} f_{il} = e_{ik} \qquad \forall i \in P, k \in N^- \tag{4.12}$$

$$y_{O^+} = 0 \tag{4.13}$$

$$y_l + \sum_{i \in P} f_{il} \leq y_s + Q_v(1 - x_{ls}^v) \qquad \forall v \in M, l \in N^+, s : (l,s) \in E \tag{4.14}$$

$$y_k - \sum_{i \in P} \sum_{l \in N_k^+} f_{il} \leq y_s + Q_v(1 - x_{ks}^v) \qquad \forall v \in M, k \in N^-, s : (k,s) \in E \tag{4.15}$$

$$y_{O^-} = 0 \tag{4.16}$$

$$\sum_{v \in M} Q_v z_k^v \geq y_k \qquad \forall k \in N^- \tag{4.17}$$

$$\sum_{v \in \mathcal{M}} Q_v g_l^v \geq y_l \qquad \forall l \in N^+ \tag{4.18}$$

$$D_{O_+} = 0 \tag{4.19}$$

$$D_l + t_{ls} + w_{ls} \leq D_s + \overline{M}(1 - x_{ls}^v) \quad \forall v \in M, (l,s) \in E \setminus \{(l, O^-) : l \in O^+ \cup N^-\} \tag{4.20}$$

$$D_k + t_{kO^-} + w_{kO^-} \leq D_{O^-}^v + \overline{M}(1 - x_{kO^-}^v) \qquad \forall v \in M, k \in O^+ \cup N^- \tag{4.21}$$

$$D_l \leq D_k \qquad \forall k \in N^-, l \in N_k^+ \tag{4.22}$$

$$\underline{D}_l \leq D_l \leq \overline{D}_l \qquad \forall l \in V \tag{4.23}$$

$$\underline{D}_{O^-} \leq D_{O^-}^v \leq \overline{D}_{O^-} \qquad \forall v \in M \tag{4.24}$$

$$z_k^v \in \{0,1\} \qquad \forall k \in N^-, v \in M \tag{4.25}$$

$$g_l^v \in \{0,1\} \qquad \forall l \in N^+, v \in M \tag{4.26}$$

$$x_{ls}^v \in \{0,1\} \qquad \forall (l,s) \in E, v \in M \tag{4.27}$$

$$f_{il} \geq 0 \qquad \forall i \in P, l \in N^+ \tag{4.28}$$

$$y_l \geq 0 \qquad \forall l \in V \cup O \tag{4.29}$$

$$D_l \geq 0 \qquad \forall l \in V \tag{4.30}$$

$$D_{O^-}^v \geq 0 \qquad \forall v \in M \tag{4.31}$$

$$\tag{4.32}$$

$\overline{U}^{il} = \min\{p_{ijd}, l \in NF_j^+ \text{ for some } j \in Fa; e_{ikd}\}$

$\overline{M} = M \times 3600$ with $M$ as the maximum number of consecutive hours of driving.

Constraints (4.2) impose that each customer is served by a single vehicle. Constraints (4.3) state that if a vehicle does not visit a customer then it cannot visit any of its associated replicas. Constraints (4.4) and (4.5) ensure that if a replica or a customer is visited, respectively, the vehicle leaves that location once. Otherwise, the vehicle does not leave the location. Constraints (4.6) establish that if a vehicle leaves a location then it must also arrive there. Otherwise, the vehicle does not visit that location. Constraints (4.7) and (4.8) make sure that each vehicle starts and ends at the correct location, taking into account that after leaving the depot a vehicle will visit a replica, and that it will arrive to its final destination coming from a customer, or it will go from the initial position to the final position if the vehicle is not used. Constraints (4.9) establish that if a vehicle visits a customer and a customer's replica is not visited then this replica does not supply any product. Constraints (4.10) establish that if there is stock of a product then the demand of that product should be satisfied with it as much as possible. In other words, these constraints give preference to stocks. Constraints (4.11) state that, for each farmer, the amount of a product to be delivered does not exceed the production. Constraints (4.12) state that, for each customer, demands are satisfied. Constraints (4.14) and (4.15) give the load of a vehicle when it arrives at a location. Constraints (4.13) and (4.16) state that each vehicle leaves and arrives empty at the depot. Constraints (4.17) and (4.18) ensure that the capacity of each vehicle is not exceeded. Constraints (4.20) and (4.21) are time elapsed constraints. Splitting such constraints is necessary because, when regarding the final depot, which is a location shared by all routes, each vehicle will have its arriving time. The arrival time at any other location is unique as it is only allowed to be visited once. These constraints eliminate subtours, as the same location cannot have different arriving times. Constraint (4.19) initializes the time elapsed at the initial position. Constraints (4.22) state the precedences between customers and the associated replicas. Constraints (4.23) to (4.24) ensure that time

windows are not violated. Constraints (4.25) to (4.33) state the variable requirements.

The linear objective function (4.1) minimizes the total cost of transportation. This cost has two components: the travel cost and a service cost. Service cost is considered in order to avoid visiting replicas unnecessarily, this is, visiting replicas where nothing is picked up. Note that, without this cost, the model allows optimal solutions like the following one: a given vehicle visits two customers after visiting three replicas, two replicas of the same farmer (one for each customer) and then a replica of other farmer (for one of the customers); the vehicle loads at the former and the latter and simply visits the second replica with no loading. Although travel cost is null between replicas of the same farmer, adding a positive cost for using a replica does not allow such alternative optimal solutions.

Farmers can be ordered by non-descending order of the participation ratio, and one can add constraint (4.33) for the first ordered farmers, in order to ensure that they deliver products (social responsibility).

$$\sum_{l \in NF_j^+} \sum_{v \in M} g_l^v \geq 1 \qquad \text{for some farmers } j \text{ with the worst participation ratios.} \tag{4.33}$$

# Chapter 5

# Unaggregated Approach

Using exact algorithms to solve the presented MILP model proved to be very time consuming when addressing larger instances of the problem. Due to the prohibitive long CPU times needed to obtain the optimal solution, one considered a second approach where the assignment of production to customers' demand and the determination of pickup and delivery routes are addressed separately and are both solved using heuristics. Although heuristics can not guarantee optimal solutions, they are a viable alternative since they can provide "good" feasible solutions in a reasonable amount of time. In this approach, the assignment problem is solved first in order to establish which farmers will in fact satisfy a customer's request and only then pickup and delivery routes are determined.

## 5.1 Assignment of Production to Demand

In order to solve the assignment problem, an heuristic was developed which takes into account customers' preferences and equity criteria used as a way to pursue a fair and balanced distribution of benefits amongst all stakeholders. Customers are ordered accordingly to their satisfaction ratio as mentioned in previous chapters, and farmers are ordered according to their "combined" ratio, a ratio that combines the participation ratio and a "distance" ratio. By considering this combined ratio one attempts to contribute for a balanced participation amongst farmers while simultaneously aiming towards the minimization of the total distribution cost.

**Farmer's Combined Ratio** The distance ratio gives a relative distance between a farmer and a customer, and assumes values between zero and one. The farther a farmer and a customer are from each other the larger is the ratio. Together with the participation criteria presented in chapter 3, a combined ratio, the weighted average of both ratios, is defined. The weights are the relative importance of the ratio components.

For each pair of farmer $j \in Fa$ and customer $k \in Cl$, the *distance ratio* is the following:

$$D_{jk} = \frac{d_{jk}}{max_{l \in Fa}\{d_{lk}\}}.$$

For each delivery day $d \in W_D$, the *combined ratio* $R_{jk}$ is

$$R_{jk} = \lambda_1 Hf_j + \lambda_2 D_{jk} \qquad \lambda_1 + \lambda_2 = 1, \;\; \lambda_1, \lambda_2 \geq 0.$$

This ratio also assumes values between zero and one. For the same value of participation ratio, the farther a farmer and a customer are from each other the larger is the combined ratio. In the same way, for the same values of distance ratio, the larger the participation ratio, the larger is the combined ratio.

**Assignment procedure** The assignment problem is solved by the use of a heuristic which assigns production to customers' demand, taking into account customers' preferences and equity criteria. Customers' preferences refer to the fact that some customers prefer not to receive a product's request if it cannot be fully satisfied while customers' priorities are established by their satisfaction ratio. Farmers' priorities are established by their combined ratio. The heuristic also enforces that an order for a storable product is to be satisfied first and foremost by products in stock (Algorithm 5), and only then by farmers.

Two different procedures of assigning production to customers' demand are presented. In the first procedure (Algorithm 7), the assignment process only takes into account farmers' priorities. The second procedure (Algorithm 6) is an attempt to reduce the number of farmers that supply a customer. In this procedure, if a selected farmer is not able to fully satisfy a product's request, then the next farmer is inspected. The assignment procedure is described in Algorithm 4.

The following additional notation is necessary.

For a given day $d \in D_W$:

- $PCl_{kd}$ - set of products $i \in P$ requested by customer $k \in Cle_d$, $PCl_{kd} = \{i \in P : q_{ikd} > 0\}$.
- $PS_d$ - set of products $i \in Ps$ in stock, $PS_d = \{i \in Ps : pa_{id} > 0\}$.

The output of the assignment procedure is the four-dimensional matrix $[e_{ijkd}]$.

## 5.2 Distribution Routes Planning

In the previous section the assignment of production to customers' demand is established, therefore determining which farmers will supply each customer with which products and the corresponding quantities. Afterwards it is necessary to obtain the pickup and delivery distribution routes. A two stage heuristics

**Algorithm 4** Assignment procedure, for a given day $d$

**for all** $k \in Cl_d^{ord}$ **do**
  $Fa^{ord} \leftarrow$ set of farmers of $Fap_d$ ordered by non-descending order of their combined ratio
  **for all** $i \in PCl_{kd}$ **do**
    **if** $k \in Cl_{tot}$ **then** ▷ customer with preferences
      **if** $i \in PS_d$ **then** ▷ storable product
        **if** $q_{ikd} \leq \sum_{j \in Fap_d} p_{ijd} + pa_{id}$ **then**
          *fromWarehouse()* ▷ assign to the warehouse
          **if** $q_{ikd} > 0$ **then**
            *fromFarmers1()* or *fromFarmers2()* ▷ assign to farmers
          **end if**
        **else**
          $e_{ijkd} \leftarrow 0, \forall j \in Fap_d$, $e_{i,AR,kd} \leftarrow 0$
        **end if**
      **else** ▷ fresh product
        **if** $q_{ikd} \leq \sum_{j \in Fap_d} p_{ijd}$ **then**
          *fromFarmers1()* or *fromFarmers2()* ▷ assign to farmers
        **else**
          $e_{ijkd} \leftarrow 0, \forall j \in Fap_d$
        **end if**
      **end if**
    **else** ▷ customer without preferences
      **if** $i \in PS_d$ **then** ▷ storable product
        *fromWarehouse()* ▷ assign to the warehouse
        **if** $q_{ikd} > 0$ **then**
          *fromFarmers1()* or *fromFarmers2()* ▷ assign to farmers
        **else**
          $e_{ijkd} \leftarrow 0, \forall j \in Fap_d$
        **end if**
      **else** ▷ fresh products
        $e_{i,AR,kd} \leftarrow 0$
        *fromFarmers1()* or *fromFarmers2()* ▷ assign to farmers
      **end if**
    **end if**
  **end for**
**end for**
**return** $[e_{ijkd}]$

**Algorithm 5** $fromWarehouse()$

$e_{i,AR,kd} \leftarrow min\{q_{ikd}, pa_{id}\}$
$q_{ikd} \leftarrow q_{ikd} - e_{i,AR,kd}$
$pa_{id} \leftarrow pa_{id} - e_{i,AR,kd}$
**if** $pa_{id} = 0$ **then**
    $PS_d \leftarrow PS_d - \{i\}$
**end if**

**Algorithm 6** $fromFarmers2()$

**for all** $j \in Fa^{ord}$ **do**
    **if** $p_{ijd} \geq q_{ikd}$ **then**
        $e_{ijkd} \leftarrow q_{ikd}$
        $q_{ikd} \leftarrow 0$
        $p_{ijd} \leftarrow p_{ijd} - e_{ijkd}$
    **else**
        **if** $j \neq |Fa^{ord}|$ **then**
            **if** $p_{i,j+1,d} \geq q_{ikd}$ **then**
                $e_{i,j+1,kd} \leftarrow q_{ikd}$
                $q_{ikd} \leftarrow 0$
                $p_{i,j+1,d} \leftarrow p_{i,j+1,d} - e_{i,j+1,kd}$
            **else**
                $e_{ijkd} \leftarrow p_{ijd}$
                $q_{ikd} \leftarrow q_{ikd} - e_{ijkd}$
                $p_{ijd} \leftarrow 0$
            **end if**
        **else**
            $e_{ijkd} \leftarrow p_{ijd}$
            $q_{ikd} \leftarrow q_{ikd} - e_{ijkd}$
            $p_{ijd} \leftarrow 0$
        **end if**
    **end if**
**end for**

**Algorithm 7** $fromFarmers1()$

1: **for all** $j \in Fa^{ord}$ **do**
2: $e_{ijkd} \leftarrow min\{q_{ikd}, p_{ijd}\}$
3: $q_{ikd} \leftarrow q_{ikd} - e_{ijkd}$
4: $p_{ijd} \leftarrow p_{ijd} - e_{ijkd}$
5: **end for**

$$\begin{bmatrix} 0 & 7 & 9.5 & 3 & 3 & 4 \\ 7 & 0 & 5.5 & 16 & 16 & 19 \\ 9.5 & 8.5 & 0 & 20 & 20 & 19 \\ 3 & 12 & 13 & 0 & 0 & 1.5 \\ 3 & 12 & 13 & 0 & 0 & 1.5 \\ 4 & 10 & 13 & 1.5 & 1.5 & 0 \end{bmatrix}$$

Distance matrix (Km)

$$\begin{bmatrix} 0 & 19 & 24 & 8 & 8 & 13 \\ 20 & 0 & 12 & 25 & 25 & 29 \\ 25 & 12 & 0 & 20 & 20 & 29 \\ 9 & 31 & 34 & 0 & 0 & 7 \\ 9 & 31 & 34 & 0 & 0 & 7 \\ 18 & 44 & 43 & 7 & 7 & 0 \end{bmatrix}$$

Time matrix (minutes)

Figure 5.1: Example - Distance and time matrices.

was implemented, in which the first phase is a constructive heuristic based on *Clarke and Wright's Savings Algorithm* (CWSA) for the classical vehicle routing problem (VRP) presented in 1964 [12]. Two different versions are presented: a parallel and a criterion based version. This constructive phase is followed by an improvement phase where two local search procedures are performed consecutively, both guided by a simulated annealing algorithm. These local search procedures attempt to swap nodes within an establish route in order to improve upon the solution given by the constructive procedure.

Throughout the rest of the chapter, different figures will be presented as examples of some the different implemented methods. Data from the smallest tested instance (instance_1) is presented. This instance has 3 customers ($Cl\_1, Cl\_2, Cl\_3$) and 3 suppliers ($W, F\_1, F\_2$), where $W$ represents the warehouse. Note that the warehouse $W$ and farmer $F\_1$ are the same geographical place. Distance and time matrices are $n \times n$ where $n$ equals the sum of customers and suppliers (see Figure 5.1). Lines and columns are ordered in the following way: $[Cl\_1, Cl\_2, Cl\_3, W, F\_1, F\_2]$. Time windows for both customers and farmers are also presented in Table 5.1. customers' products requests and suppliers available production are presented in Table 5.3. Table 5.4 presents the solution obtained from the production assignment method.

Note that throughout the heuristic's explanation we define a *Route* entity $R$ by its ordered set of nodes (its route $r^R$) but it also possesses several other proprieties associated with it namely:

- $D_i^R$ - distance traveled by vehicle when it arrives at location $i$ in *Route* $R$

| Customer | Lower limit | Upper limit |
|---|---|---|
| 1 | 0 | 240 |
| 2 | 0 | 240 |
| 3 | 0 | 240 |

| Supplier | Lower limit | Upper limit |
|---|---|---|
| W | 0 | 0 |
| F_1 | 0 | 480 |
| F_2 | 0 | 480 |

Table 5.1: Time windows (minutes).

| Customer | Product | Quantity (Kg) |
|---|---|---|
| 1 | 1 | 3 |
| 1 | 2 | 100 |
| 1 | 4 | 20 |
| 1 | 5 | 15 |
| 2 | 2 | 5 |
| 2 | 15 | 1 |
| 3 | 2 | 100 |
| 3 | 4 | 5 |
| 3 | 5 | 15 |
| 3 | 15 | 1 |

Table 5.2: Customers' total demand.

| Supplier | Product | Quantity (Kg) |
|---|---|---|
| W | 2 | 20 |
| $F\_1$ | 2 | 2000 |
| $F\_1$ | 5 | 10000 |
| $F\_2$ | 1 | 3 |
| $F\_2$ | 4 | 25 |
| $F\_2$ | 14 | 3 |
| $F\_2$ | 15 | 25 |

Table 5.3: Production yields.

- $T_i^R$ - elapsed time when a vehicle arrives at location $i$ in *Route R*
- $y_i^R$ - load of a vehicle when it arrives at location $i$ in *Route R*
- $Cl^R$ - set of clients visited in *Route R* (ordered accordingly with its position in the route)
- $Fa^R$ - set of farmers visited in *Route R* (ordered accordingly with its position in the route). Note that a farmer may be visited more than once in the same route
- $\delta$ - resulting saving if the route is obtained by the merge of two existing routes.

These proprieties are updated through the *buildRoute*$(r, \delta)$ method (see Algorithm 8).

Some of the notation used in this section has already been defined in previous chapters. The additional notation used is:

- $S$ - set of routes which compose the final solution, $S = \{S_1, S_2, ..., S_{|S|}\}$
- $r_j^R$ - set of ordered nodes $j$, $j = 1, ..., |r^R|$ which compose a *Route R*
- Problem characteristics lead to: $\forall R \in S,\ r_1^R = r_{|r^R|}^R = O,\ r_{(|r^R|-1)}^R \in Cl$

| Customer | Product | Supplier | Quantity (Kg) |
|---|---|---|---|
| 1 | 1 | $F\_2$ | 3 |
| 1 | 2 | $W$ | 20 |
| 1 | 2 | $F\_1$ | 80 |
| 1 | 4 | $F\_2$ | 20 |
| 1 | 5 | $F\_2$ | 15 |
| 2 | 2 | $F\_1$ | 5 |
| 2 | 15 | $F\_2$ | 1 |
| 3 | 2 | $F\_1$ | 100 |
| 3 | 4 | $F\_2$ | 5 |
| 3 | 5 | $F\_1$ | 15 |
| 3 | 15 | $F\_1$ | 1 |

Table 5.4: Assignment production to requests.

**Algorithm 8** *buildRoute*$(r, \delta)$

1: $route \leftarrow r$
2: Update $D^R$ ▷ update distances
3: Update $T^R$ ▷ update times
4: Update $y^R$ ▷ update loads
5: Update $Cl^R$ ▷ update customers' vector
6: Update $Fa^R$ ▷ update suppliers' vector
7: $saving \leftarrow \delta$ ▷ saving
8: **return** *Route*

- $\delta$ - savings
- $\mathcal{I}_{Y,Z}$ - set of nodes in the intersection of *Route* $Y$ and $Z$. $\mathcal{I}_{Y,Z} = r^Y \cap r^Z \setminus \{O\} : Y, Z \in S$
- $d_{ij}$ - distance between locations $i$ and $j$
- $t_{ij}$ - travel time between locations $i$ and $j$
- $\underline{D}_i$ - minimum arriving time at location $i$
- $\overline{D}_i$ - maximum arriving time at location $i$
- $\bar{B}$ - set of blocks
- $B$ - block
- $N$ - set of nodes to be merged
- $Q$ - vehicles' capacity (homogeneous fleet)
- $\alpha$ - total distance of edges to be removed
- $\beta$ - total distance of edges to be inserted

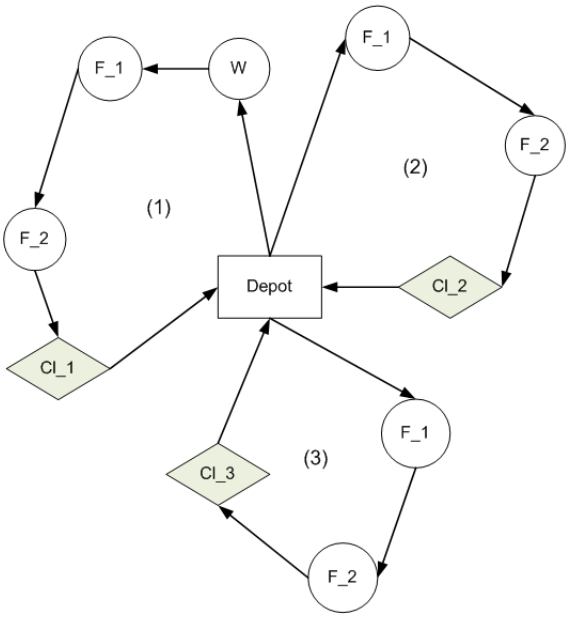
The distribution main procedure is presented in Algorithm 9.

**Algorithm 9** *Main*

1: $S = \emptyset$
2: $S \leftarrow Initialize_routes()$ ▷ initialization
3: **while** (solution improves) $\wedge$ $|S| > 1$ **do** ▷ constructive phase
4: $S \leftarrow ACWSA(S)$ ▷ parallel or criterion based algorithm
5: **end while**
6: $S \leftarrow NodeSwap(S)$ ▷ improvement phase

### 5.2.1 Construction Phase

An initial solution is obtained by constructing a route for each customer using the *Nearest Neighbor Algorithm* where precedence constraints are assured, this is, all farmers that supply a customer are visited before the customer in the order given by the *Nearest Neighbor Algorithm* while respecting capacity and time windows constraints. The initial solution obtained is presented in Figure (5.2).




**Route (1)**

| | | | | | | |
|---|---|---|---|---|---|---|
| Route – | [ depot | W | F_1 | F_2 | Cl_1 | depot ] |
| Distance – | [ 0 | 0 | 0 | 1.5 | 5.5 | 8.5 ] |
| Time – | [ 0 | 0 | 10 | 27 | 55 | 73 ] |
| Load - | [ 0 | 0 | 20 | 115 | 138 | 0 ] |

**Route (2)**

| | | | | | |
|---|---|---|---|---|---|
| Route – | [ depot | F_1 | F_2 | Cl_2 | depot ] |
| Distance – | [ 0 | 0 | 1.5 | 11.5 | 27.5 ] |
| Time – | [ 0 | 0 | 17 | 71 | 106 ] |
| Load - | [ 0 | 0 | 5 | 6 | 0 ] |

**Route (3)**

| | | | | | |
|---|---|---|---|---|---|
| Route – | [ depot | F_1 | F_2 | Cl_3 | depot ] |
| Distance – | [ 0 | 0 | 1.5 | 14.5 | 34.5 ] |
| Time – | [ 0 | 0 | 17 | 70 | 100 ] |
| Load - | [ 0 | 0 | 115 | 121 | 0 ] |

Figure 5.2: Example - Initial solution.

After obtaining an initial solution, a constructive procedure attempts to merge routes, so that the resulting route is feasible. This procedures is based on the Clarke and Wright Savings Algorithm (CWSA) for the classical VRP problem and from this point forward we will address it as *Adapted Clarke and Wright Savings Algorithm* (ACWSA). Two different versions of this algorithm were implemented: a parallel version (see Algorithm 10), similar to the parallel version of the CWSA, and a criterion based version as presented in Algorithm 11).

**Parallel ACWSA** The presented parallel version of ACWSA is very similar to the parallel version of the classical CWSA. In the presented procedure we determine the savings matrix resulting of merging all routes between themselves and choose the one which yields the greatest saving. As merging routes is not as simple as in the original CWSA, the usual savings matrix is substituted by a merged routes matrix where an entire

*Route* entity is stored, with all its associated characteristics. Then we chose the resulting *Route* $R$ which yields the greatest saving and if $max\{\delta^R\} > 0$, we update solution's $S$ route pool by removing the original routes and adding the resulting merged route. If $max\{\delta^R\} \leq 0$ the procedure ends. (See Algorithm 10).

**Algorithm 10** Parallel $ACWSA(S)$

**for all** $i = 1..|S|$ **do**
  **for all** $j = 1..|S|$ **do**
    **if** $i \neq j$ **then**
      $Y \leftarrow S_i$
      $Z \leftarrow S_j$
      $mergedRoute = mergeSaving(Y, Z)$ ▷ determine merge
      $mergeSavings[i][j] \leftarrow mergedRoute$ ▷ update merge routes matrix
    **end if**
  **end for**
**end for**
$\Delta \leftarrow \delta^R : \delta^R = max\{\delta^X : X \in mergeSavings\}$ ▷ select *Route* with the highest saving
**if** $\Delta > 0$ **then**
  $S \leftarrow S \setminus \{Y, Z\}$ ▷ update solution *Route* pool
  $S \leftarrow S \cup \{R\}$
**end if**
**return** $S$

**Criterion based ACWSA** In the criterion based version two routes are chosen from the current solution $S$ according to some criteria. The criterion used in this work was to choose two routes which would have the least load when arriving at the last client served. The objective was to promote the most possible feasible merges taking into account capacity constraints. Other criteria can also be considered, like choosing the two closest customers which are visited in different routes. (See Algorithm 11).

**Algorithm 11** Criterion based $ACWSA(\mathcal{S})$

*Pick two routes* $Y$ *and* $Z : Y, Z \in S$ *according to some criteria*
$mergedRoute1 = mergeSaving(Y, Z)$
$mergedRoute2 = mergeSaving(Z, Y)$
$\Delta = \delta^R : \delta^R = max\{\delta^{mergedRoute1}, \delta^{mergedRoute2}\}$ ▷ select *Route* with the highest saving
**if** $\Delta > 0$ **then**
  $S \leftarrow S \setminus \{Y, Z\}$ ▷ update solution *Route* pool
  $S \leftarrow S \cup \{R\}$
**end if**

As we have an asymmetric graph and precedence constraints, we must assure that after every merge, the resulting route maintains the current orientation and feasible precedences. For each possible merge between two routes we may have two different results depending on how those routes are merged. It is established that one route must remain unchanged while we attempt to merge with the second route, this is, the second route needs to adapt to the first one. These routes will be addressed as dominated and dominant routes whether they need to adapt or not, respectively. Note that the order of the inputs (*Routes*) determines whether the route is dominant or dominated. The first input will be established as dominant and the second input as dominated (see Algorithm 12).

Two different procedures were implemented in order to take into account the existence of common farmers in both routes.

**Algorithm 12** $mergeSaving(Y, Z)$

```
Determine intersection vector I_{Y,Z} between Routes Y and Z
if I_{Y,Z} ≠ ∅ then
    mergedRoute = deltaIntersect(Y, Z)
else
    mergedRoute = deltaNoIntersect(Y, Z)
end if
return mergedRoute
```

If there is no intersection $\mathcal{I}_{Y,Z} = \emptyset$, then *Routes* $Y$ and $Z$ are merged sequentially, this is, simply connect the last customer from the dominant *Route* to the first farmer to be visited of the dominated one as described Algorithm 13. In this case it only makes sense to verify if time windows constraints hold for every node in the dominated *Route* as capacity constraints will always be respected because a vehicle will certainly leave empty from a *Routes*'s last client.

If some farmers supply customers in both routes, this is, the intersection is not empty, the procedure tries to eliminate the need to visit a common farmer more than once, by transferring its load from the dominated *Route* to the dominant one. In order to verify which farmers are in fact able to be merged, an auxiliary function $Operator_1$ (Algorithm 15), was developed in order to establish which farmers can in fact be merged( set $N$). If we cannot merge any farmers then routes are merged sequentially as explained before. If merges are possible then farmer nodes belonging to $N$ are merged into the dominant route while the remaining nodes yet to be visited in the dominated *Route* are connected to the dominant route keeping the order in which they were visited, as depicted in Figure 5.3.

As previously mentioned, if the intersection is not empty, then $Operator_1$ is preformed, determining this way, which farmer nodes are in fact able to be merged. This allows for loads to be transfered from the dominated *Route* to the dominant *Route*, therefore avoiding to visit the same farmer once more in the same route. This method also calculates set $\bar{B}$ which contains blocks of nodes taking into account intersection

characteristics. A block is formed by the last farmer not present in the intersection, the consecutive nodes present in the intersection and the next node not present in the intersection. These blocks provide a useful index pool for calculating savings in an efficient way. Note that $\bar{B}$ will only contain blocks which represent a feasible merge, regarding capacity constraints.

The *capacity*$(s)$ procedure presented in Algorithm 16 verifies merging feasibility regarding capacity constraints. It verifies, for all customers which are to be visited after farmer $j$ in the dominant *Route*, if the resulting load after merging a farmer node is less or equal to the capacity of the vehicle. The output $x = 0$ means capacity feasibility while $x = -1$ means that the merge is infeasible.

Likewise, the *timeWindows* procedure presented in Algorithm 17 verifies merging feasibility regarding time windows constraints.

The $\alpha$ parameter stands for the total distance regarding nodes to be deleted, and it is calculated using the $\bar{B}$ set (see Algorithm 18). The $\beta$ parameter stands for the total distance regarding edges to be inserted, and it is calculated using the $\bar{B}$ set (see Algorithm 19).

**Algorithm 13** *deltaNoIntersect*$(Y, Z)$

$r \leftarrow \emptyset$
$x \leftarrow timeWindows(Y, Z)$
**if** $x \neq -1$ **then** ▷ if it is feasible
  $r^Y \leftarrow r^Y \setminus \{r^Y_{|r^Y|}\}$ ▷ remove end location from dominant *Route*
  $r^Z \leftarrow r^Z \setminus \{r^Z_1\}$ ▷ remove start location from dominated *Route*
  $c \leftarrow Cl^Y_{|Cl^Y|}$ ▷ last visited customer in dominant *Route*
  $r \leftarrow r^Y \cup r^Z$ ▷ merge routes
  $\delta \leftarrow d_{cO^+} + d_{O^+ r^Z_1} - d_{cr^Z_1}$
  $mergedRoute = buildRoute(r, \delta)$ ▷ builds resulting *Route*
**else** ▷ if it is not feasible
  $\delta = x$
  $mergedRoute = buildRoute(r, \delta)$ ▷ builds resulting "empty" *Route*
**end if**
**return** *mergedRoute*

**Algorithm 14** *deltaIntersect*$(Y, Z)$

$[N, \bar{B}] \leftarrow Operator_1(Y, Z)$
**if** $N = \emptyset$ **then** ▷ if no farmers can be merged
    $mergedRoute = deltaNoIntersect(Y, Z)$
**else**
    $r = \emptyset$
    $r^{\hat{Z}} \leftarrow r^Z \setminus \{N\}$ ▷ remove from dominated *Route* locations that are no longer visited - cloned vector
    $x \leftarrow timeWindows(Y, r^{\hat{Z}})$
    **if** $x \neq -1$ **then** ▷ it is feasible
        $r^Y \leftarrow r^Y \setminus \{r^Y_{|r^Y|}\}$ ▷ remove end location from dominant *Route*
        $r^{\hat{Z}} \leftarrow r^{\hat{Z}} \setminus \{r^{\hat{Z}_1}\}$ ▷ remove start location from dominated *Route* (clone)
        $r \leftarrow r^Y \cup r^{\hat{Z}}$ ▷ merge routes
        $\alpha = getAlpha(\bar{B})$
        $\beta = getBeta(\bar{B})$
        $c \leftarrow Cl^Y_{|Cl^Y|}$ ▷ last visited customer in dominant *Route*
        $\delta \leftarrow d_{cO^+} + d_{O^+Z_2} + \alpha - \beta$
        $mergeRoute = buildRoute(r, \delta)$
    **else** ▷ if it is not feasible
        $\delta = x$
        $mergeRoute = buildRoute(r, \delta)$ ▷ builds resulting "empty" *Route*
    **end if**
**end if**
**return** *mergedRoute*

**Algorithm 15** $Operator_1(Y, Z)$

1: $s \leftarrow 2$ ▷ the start position is not verified
2: **while** $s < |r^Z|$ **do**
3: **if** $r_s^Z \in \mathcal{I}_{Y,Z}$ **then** ▷ if the farmer supplies both routes
4: $x \leftarrow capacity(s)$ ▷ verifies capacity constraints
5: **if** $x \neq -1$ **then** ▷ it is feasible
6: $B = \emptyset$
7: $B \leftarrow \{r_{s-1}^Z, r_s^Z\}$
8: $N \leftarrow \{r_s^Z\}$ ▷ update nodes to be merged
9: $s \leftarrow s + 1$
10: **while** $r_s^Z \in \mathcal{I}_{Y,Z} \quad \wedge \quad x \neq -1$ **do** ▷ if the farmer supplies both routes
11: $x \leftarrow capacity(s)$ ▷ verifies capacity constraints
12: **if** $x \neq -1$ **then**
13: $B \leftarrow B \cup \{r_s^Z\}$
14: $N \leftarrow N \cup \{r_s^Z\}$ ▷ update nodes to be merged
15: $s \leftarrow s + 1$
16: **end if**
17: **end while**
18: $B \leftarrow B \cup \{r_s^Z\}$
19: $\bar{B} \leftarrow \bar{B} \cup \{B\}$ ▷ updates $\bar{B}$ set
20: **end if**
21: **end if**
22: $s \leftarrow s + 1$
23: **end while**
24: **return** $[N, \bar{B}]$

**Algorithm 16** *capacity*$(s)$

let $j$ be the farmer in position $s$ in the dominated *Route* $Z$
let $l$ be the position of the last visit to farmer $j$ in the dominant *Route* $Y$
$x = 0$
**for all** $k \in Cl^Y$ **do**
    **if** $x \neq -1$ **then**
        $a \leftarrow Cl^Y_k$
        let $i$ be the position of customer $a$ in the dominant *Route* $Y$
        **if** $i > l$ **then**
            $w = y^Y_i + (y^Z_{s+1} - y^Z_s)$
            **if** $w > Q$ **then**
                $x = -1$
            **end if**
        **end if**
    **end if**
**end for**
**return** $x$

**Algorithm 17** *timeWindows*$(Y, Z)$

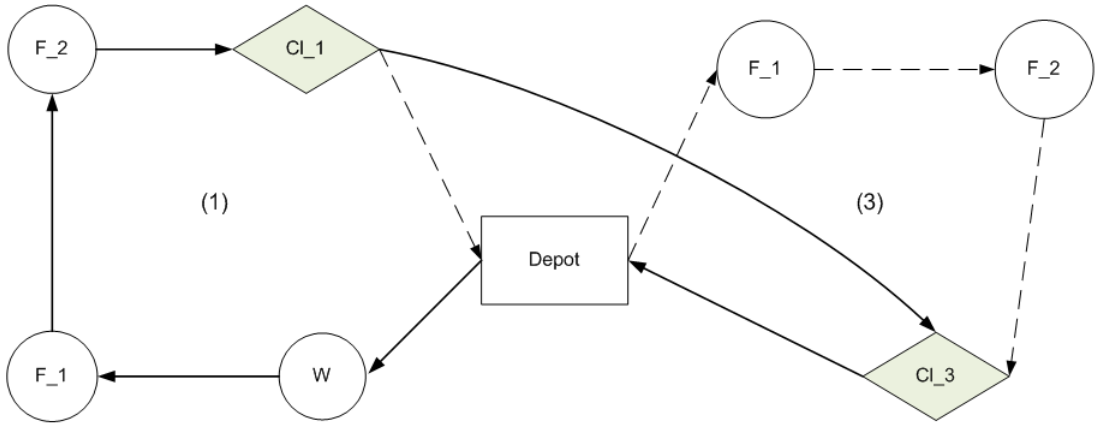
$x = 0$
$currentNode \leftarrow Cl^Y_{|Cl^Y|}$ ▷ last visited customer in dominant *Route*
**for all** $k \in r^Z \setminus \{r^Z_1\}$ **do** ▷ for all dominated *Route*'s locations aside from the start position
    **if** $x \neq -1$ **then**
        $j \leftarrow r^Z_k$
        $D = D_{currentNode} + t_{currentNode,j} + serviceTime$
        **if** $D < \underline{D}_j$ **then**
            $D = \underline{D}_j$ ▷ a vehicle must wait if it arrives early
            $currentNode \leftarrow j$
        **end if**
        **if** $D > \overline{D}_j$ **then**
            $x \leftarrow -1$ ▷ it is infeasible
        **else**
            $currentNode \leftarrow j$
        **end if**
    **end if**
**end for**
**return** $x$

**Algorithm 18** $getAlpha(\bar{B})$

$\alpha = 0$
**for all** $i \in \bar{B}$ **do**
    $B = \bar{B}_i$
    **while** $j < |B|$ **do**
        $\alpha \leftarrow \alpha + d_{B_j B_{j+1}}$
        $j \leftarrow j + 1$
    **end while**
**end for**
**return** $\alpha$

**Algorithm 19** $getBeta(\bar{B})$

$\beta = 0$
$c \leftarrow Cl^Y_{|Cl^Y|}$ ▷ last visited customer in dominant *Route*
$s \leftarrow B_{|B|}$
**for all** $i \in \bar{B}$ **do**
    $B \leftarrow \bar{B}_i$
    **if** $i = 1$ **then**
        **if** $B_1 = O^+$ **then** ▷ if the first node of a block is the start position
            $\beta \leftarrow \beta + d_{cs}$
        **else**
            $\beta \leftarrow \beta + d_{cr_1^Z} + d_{B_1 s}$
        **end if**
    **end if**
    $\beta \leftarrow \beta + d_{B_1 s}$
**end for**
**return** $\beta$




**I** = [ F_1 F_2 ]

**N** = [ F_1 F_2 ]

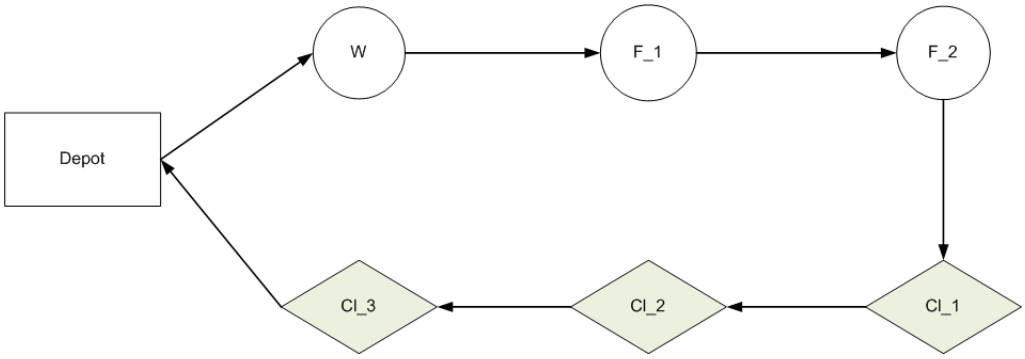
**B** = [ depot F_1 F_2 Cl_3 ]

**Alpha** = 14.5
**Beta** = 9.5

**Delta** = 3 + 14.5 - 9.5 = 8

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Route** | [ depot | W | F_1 | F_2 | Cl_1 | Cl_3 | depot ] |
| **Distance** | [ 0 | 0 | 0 | 1.5 | 5.5 | 15 | 35 ] |
| **Time** | [ 0 | 0 | 10 | 27 | 55 | 89 | 119 ] |
| **Load** | [ 0 | 0 | 20 | 230 | 259 | 121 | 0 ] |

Figure 5.3: Example - Route merge when farmers supply both routes.




| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Route** | [ depot | W | F_1 | F_2 | Cl_1 | Cl_2 | Cl_3 | depot ] |
| **Distance** | [ 0 | 0 | 0 | 1.5 | 5.5 | 12.5 | 18 | 38 ] |
| **Time** | [ 0 | 0 | 10 | 27 | 55 | 84 | 106 | 136 ] |
| **Load** | [ 0 | 0 | 20 | 235 | 265 | 127 | 121 | 0 ] |

Figure 5.4: Example - Solution after construction phase.

### 5.2.2 Improvement Phase

After completing the constructive phase, using one of the presented *ACWSA*, a combined local search method is then applied in order to attempt to improve upon the current solution by swapping nodes within previously established *Routes* $R \in \mathcal{S}$. This procedure is divided into two phases which are executed in turns for a given pre-established number of iterations. Both phases can be regarded as specific local search procedures on their own. This composite local search method was designed so that precedence and capacity constraints could be addressed as node swaps were made. The first phase attempts to swap farmers which are visited consecutively, this is, which are visited between two non farmer nodes and in the second phase the procedure attempts to move a customer node forward one position in a route. In order to attempt to escape local optima, a simulated annealing meta-heuristic [1] was implemented in order to guide both local search methods.

**Algorithm 20** $NodeSwap(S)$

1: **for** $it = 1...IT$ **do** ▷ IT is the number of iterations (chosen by the administrator)
2: $S \leftarrow sameSetSA(S)$
3: $S \leftarrow frogLeapSA(S)$
4: **end for**
5: **return** $S$

**First phase** In the first phase, farmer nodes are swapped between themselves within a given route. In order to maintain feasibility, regarding capacity as well as precedence constraints, it is established that node swaps may only be preformed withing specific sets of farmers. Let $T$ be defined as the ordered set of consecutively visited farmers of a route which are visited between two non farmer nodes (see Algorithm 22). These sets represent the possible farmer nodes to be swapped between each other. As there might exist several set $T$ in a given route, we define $\bar{T}$ as the collection of all existing $T$ sets in that route. As node swaps are only allowed within these sets, capacity constrains as well as precedence constrains will hold. Therefore the procedure only takes into account time windows constraints in order to establish if the resulting solution is feasible or not.

**Second phase** In the second phase, we attempt to move a customer forward one position in a route. Precedence constraints hold when performing these swaps, and therefore it is only necessary to verify capacity and time constraints. This specific method was designed taking into account the following characteristic of the test case: as previously mentioned, customers are somewhat clustered in the urban area of Setúbal while farmers are scattered around the surrounding rural area. Therefore we devised a method which will try to pickup all products first, and visit customers in the end of a route. This procedure also allows for the creation of new $T$ blocks therefore expanding the search space.

| Annealing process | Simulated annealing |
|---|---|
| Used to find the thermal equilibrium of matter | Used for function optimization |
| Energy variation | Objective function variation |
| Based on temperature reduction | Based on "temperature parameter" reduction |
| State of the system | Current solution |
| Subsequent states that the system can reach | Neighborhood |
| Ground state | Optimal solution |

Table 5.5: Analogy between simulated annealing and physical annealing process.

First a greedy approach was taken, where swaps were only made if in fact they represented an improvement upon the current solution. While testing some instances we verified that it was fairly easy to get stuck in local optima and so, a simulated annealing procedure was implemented in order to guide both *sameSet* (Algorithm 23) and $frogLeap$ (Algorithm 25) methods. In this way we could hopefully escape this local optimal solutions by exploring more of the solution space.

**Simulated Annealing** Simulated annealing is a local search meta-heuristic which provides means to escape local optima by allowing moves to lower quality solutions with a pre-specified probability which is inspired by the annealing process of solids, where a solid is heated until its melting point and then it is left to cool down very slowly, in order to achieve its most regular crystal configuration (i.e. ground energy state). It is an adaptation of a special Monte Carlo method for generating sample states of a thermodynamic system which was introduced by Metropolis et al. in 1953. These states are generated in the following manner. Given a current state $i$ with energy level $E_i$

- A small perturbation is applied to state $i$ in order to generate a new state $j$ with energy level $E_j$
- If $E_j - E_i \leq 0$, then accept state $j$ as the current state
- If $E_j - E_i > 0$, then accept state $j$ with probability $p = exp(\frac{E_i - E_j}{k_b T})$ , where $T$ is the current temperature and $k_b$ is the Boltzmann Constant.

Simulated annealing uses the metropolis algorithm in order to simulate the thermal equilibrium. Some analogies can be made between the physical process and the optimization process as shown in Table 5.5

**Cooling Schedule** Simulated Annealing's performance is directly influenced by its parameters, may they be the initial or final "temperature", the number of iterations in each "temperature" or the cooling rates and functions. In order to optimize the presented procedures, tests need to be preformed using different combinations for them.

It is possible to choose from two different cooling functions each with an associated parameter. Featured cooling functions are the following:

- Geometric function - $t \leftarrow \alpha t$, where alpha is usually in the range [0.8,0.99], and the number of repetition may vary taking into account the solutions space dimensions, neighborhood dimensions or temperature schemes.
- Lundy and Mees - $t \leftarrow \frac{t}{1+\beta t}$, where $\beta$ parameter should be a very small number in order to simulate a very slow cooling process. While using this function only one iteration is preformed in each "temperature".

**Neighborhood Structure** Simulated annealing was used to guide both local search procedures and so two different neighborhood structures were devised.

For the *sameSet()* procedure a neighbor solution $S'$ is constructed by randomly selecting a *Route* $R \in S$, and then swapping two farmer nodes within a given block $T$. These neighbor solutions are only constructed if they are feasible regarding time windows constraints. If the swap is infeasible then the procedure return the selected route unchanged.

For the *frogLeap()* procedure a neighbor solution $S'$ is constructed by randomly selecting a route $R$ from the current solution $S$ and then swapping a customer node with the node that follows it (not including the last visited customer). The procedure enforces that every new solution can only be regarded if it verifies precedence, capacity and time windows constraints. If the solution resulting from the swap is infeasible then the procedure returns the selected route unchanged.

**Algorithm 21** *Simulated_Annealing*($S$)

Initialize(iTemp,fTemp,nRep, CS) ▷ initialize initial tempurature, final temperature,number of repetitions, Cooling Schedule
**while** Stoppage criteria is not met **do**
  **for** $it = 1....nRep$ **do** ▷ $nRep$ being the number of repetitions in the same temperature
    $NewRoute \leftarrow sameSet(S)$ or $frogLeap(S)$ ▷ generates new $Route$ using the appropriate local search method
    **if** $\delta^R > 0$ **then**
      $S \leftarrow S \setminus \{R\}$ ▷ update solution
      $S \leftarrow S \cup \{NewRoute\}$
    **else**
      **if** $exp(\delta^R/t) > random[0,1)$ **then**
        $S \leftarrow S \setminus \{R\}$ ▷ update solution
        $S \leftarrow S \cup \{NewRoute\}$
      **end if**
    **end if**
  **end for**
  $t \leftarrow updateTemperature()$
**end while**
**return** $S$

**Algorithm 22** $Operator_2(R)$

1: **for all** $k \in Cl^R$ **do**
2: $T = \emptyset$
3: $a = Cl_k^R$
4: let $i$ be the position of customer $a$ in *Route* $R$
5: **if** $k = 1$ **then** ▷ for the first customer to be visited in *Route* $R$
6: $s = 2$ ▷ in order to disregard the start position
7: **for** $s < i$ **do**
8: $T \leftarrow T \cup r_s^R$
9: **end for**
10: **else**
11: $b = Cl_{k-1}^R$ ▷ last customer visited before $a$
12: let $j$ be the position of customer $b$ in *Route* $R$
13: **if** $j \neq i - 1$ **then** ▷ if customers are not visited consecutively
14: $s = j + 1$ ▷ first farmer visited after customer $b$
15: **for** $s < i$ **do**
16: $T \leftarrow T \cup r_s^R$
17: **end for**
18: $\bar{T} \leftarrow \cup T$
19: **end if**
20: **end if**
21: **end for**
22: **return** $\bar{T}$

**Algorithm 23** *sameSet*$(S)$

Pick a random route $R \in S$
$Operator_2(R)$
Randomly select a block $T \in \bar{T}$
**if** $|T| \geq 2$ **then** ▷ in order to perform a swap at least 2 farmers are needed
    Randomly pick two nodes $a, b : a, b \in T$
    let $i$ and $j$ be the positions of $a$ and $b$ in *Route* $R$ ▷ let $i$ be visited before $j$ (order matters)
    $x \leftarrow timeWindowsSameSet(i, j)$
    **if** $x \neq -1$ **then**
        **if** nodes are consecutively visited **then**
            $\delta \leftarrow d_{r_{i-1}^R, r_i^R} + d_{r_i^R, r_j^R} + d_{r_j^R, r_{j+1}^R} - d_{r_{i-1}^R, r_j^R} - d_{r_j^R, r_i^R} - d_{r_i^R, r_{j+1}^R}$
        **else**
            $\delta \leftarrow d_{r_{i-1}^R, r_i^R} + d_{r_i^R, r_{i+1}^R} + d_{r_{j-1}^R, r_j^R} + d_{r_j^R, r_{j+1}^R} - d_{r_{i-1}^R, r_j^R} - d_{r_j^R, r_{i+1}^R} - d_{r_{j-1}^R, r_i^R} - d_{r_i^R, r_{j+1}^R}$
        **end if**
        $r \leftarrow r^R$ : farmers $a$ and $b$ swap positions
        $NewRoute \leftarrow buildRoute(r, \delta)$
    **else**
        $NewRoute \leftarrow R$
    **end if**
**else**
    $NewRoute \leftarrow R$
**end if**
**return** $NewRoute$

**Algorithm 24** *timeWindowsSameSet*$(i, j)$

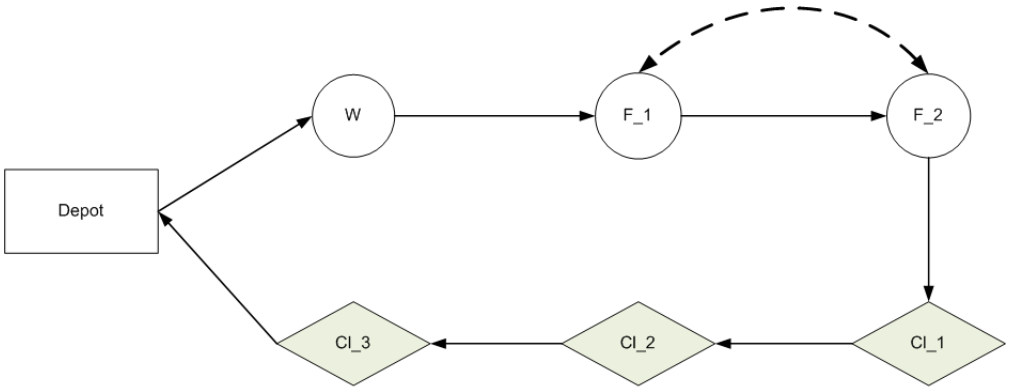
$x = 0$
$currentNode = r^R_{i-1}$ ▷ last node visited before node $r^R_i$
$D = D_{currentNode} + t_{currentNode, r^R_j} + serviceTime$ ▷ time arriving at node $r^R_j$
**if** $D \leq \underline{D}_{r^R_j}$ **then**
    $D \leftarrow \underline{D}_{r^R_j}$ ▷ a vehicle must wait if it arrives early
    $currentNode \leftarrow r^R_j$
**end if**
**if** $D > \overline{D}_{r^R_j}$ **then**
    $x \leftarrow -1$
**end if**
**if** $x \neq -1$ **then**
    $k \leftarrow i + 1$ ▷ position following farmer $a$
    **for all** $k \leq |r^R|$ **do**
        **if** $x \neq -1$ **then**
            **if** $k = j$ **then** ▷ if we are at farmer's $b$ position
                $D = D + t_{currentNode, r^R_i} + serviceTime$
                **if** $D \leq \underline{D}_{r^R_i}$ **then**
                    $D \leftarrow \underline{D}_{r^R_i}, \quad currentNode \leftarrow r^R_i$
                **end if**
                **if** $D > \overline{D}_{r^R_i}$ **then**
                    $x \leftarrow -1$
                **end if**
            **else** ▷ any other position aside form $i$ and $j$
                $D = D + t_{currentNode, r^R_k} + serviceTime$
                **if** $D \leq \underline{D}_{r^R_k}$ **then**
                    $D \leftarrow \underline{D}_{r^R_i}, \quad currentNode \leftarrow r^R_i$
                **end if**
                **if** $D > \overline{D}_{r^R_k}$ **then**
                    $x \leftarrow -1$
                **end if**
            **end if**
        **end if**
    **end for**
**end if**
**return** $x$

**Algorithm 25** $frogLeap(S)$

Pick a random route $R \in S$
Randomly pick a client node $a \in Cl^R$ except the last one
let $i$ be the position of client node $a$ in route $R$
$j = i + 1$ ▷ next visited node
$x \leftarrow capacityFrog(i, j)$
**if** $x \neq -1$ **then** ▷ it is feasible
    $x \leftarrow timeWindowsSameSet(i, j)$
**end if**
**if** $x \neq -1$ **then**
    $\delta \leftarrow d_{r_{i-1}^R, r_i^R} + d_{r_i^R, r_j^R} + d_{r_j^R, r_{j+1}^R} - d_{r_{i-1}^R, r_j^R} - d_{r_j^R, r_i^R} - d_{r_i^R, r_{j+1}^R}$
    $r \leftarrow r^R$ : nodes $a$ and $r_j^R$ swap positions
    $NewRoute \leftarrow buildRoute(r, \delta)$
**else**
    $NewRoute \leftarrow R$
**end if**
**return** $NewRoute$

**Algorithm 26** $capacityFrog(i, j)$

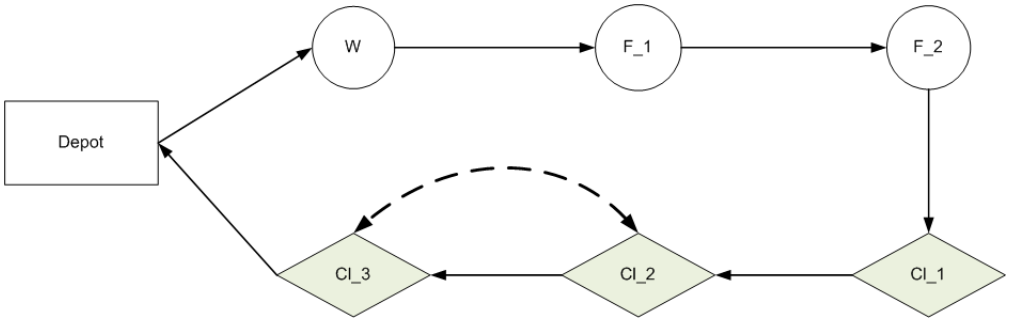
$x = 0$
$w = y_i^Y + (y_{j+1}^Y - y_j^Y)$
**if** $w > Q$ **then**
    $x \leftarrow -1$
**end if**
**return** $x$




| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Route** | [ | depot | W | F_2 | F_1 | Cl_1 | Cl_2 | Cl_3 | depot ] |
| **Distance** | [ | 0 | 0 | 1.5 | 3 | 6 | 13 | 18.5 | 38.5 ] |
| **Time** | [ | 0 | 0 | 17 | 34 | 53 | 82 | 104 | 134 ] |
| **Load** | [ | 0 | 0 | 20 | 50 | 265 | 127 | 121 | 0 ] |

**Delta** = 38 – 38.5 = -0.5

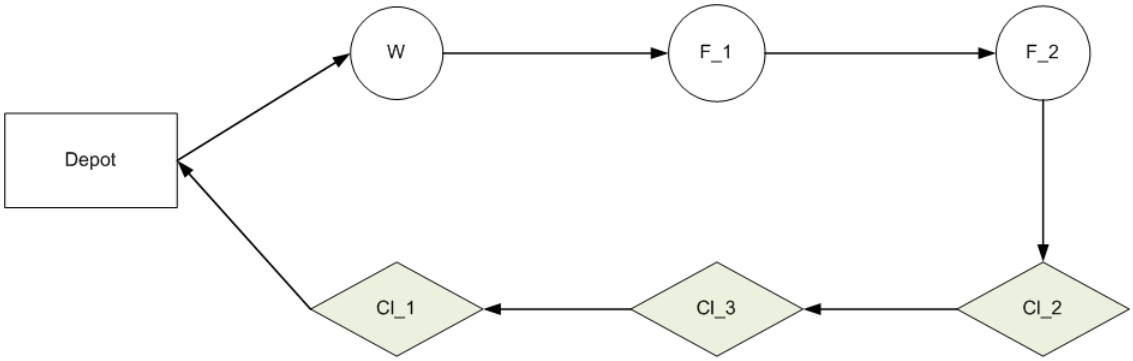
Figure 5.5: *sameSet*() - Node exchange example.




| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Route** | [ | depot | W | F_1 | F_2 | Cl_1 | Cl_3 | Cl_2 | depot ] |
| **Distance** | [ | 0 | 0 | 0 | 1.5 | 5.5 | 15 | 23.5 | 39.5 ] |
| **Time** | [ | 0 | 0 | 10 | 27 | 55 | 89 | 111 | 146 ] |
| **Load** | [ | 0 | 0 | 20 | 235 | 265 | 127 | 6 | 0 ] |

**Delta** = 38 – 39.5 = -1.5

Figure 5.6: *frogLeap*() - Node exchange example.




| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Route** | [ depot | W | F_1 | F_2 | Cl_2 | Cl_3 | Cl_1 | depot ] |
| **Distance** | [ 0 | 0 | 0 | 1.5 | 11.5 | 17 | 26.5 | 29.5 ] |
| **Time** | [ 0 | 0 | 10 | 27 | 81 | 103 | 138 | 156 ] |
| **Load** | [ 0 | 0 | 20 | 235 | 265 | 259 | 138 | 0 ] |

Figure 5.7: Example - Final solution after local search.

# Chapter 6

# Assignment of Production to the Warehouse and Final Distribution Routes Planning

After the determination of pickup and delivery routes for a delivery day (by one of the approaches), we verify if there are still storable products to be picked up during that week and if they can be brought to the warehouse, taking into account the available vehicle's capacity and the remainder available production.

After performing the last pickup and delivery route of the week, if there are still storable products to be brought to the warehouse, it is possible to determine pickup routes necessary to comply with those needs. At the end of the week, the quantity of storable products that must be available in the beginning of next week is updated.

## 6.1 Assignment of Production to the Warehouse

A simple procedure (see Algorithm 27) is proposed in order to calculate the daily quantities of storable products to pick up destined to the warehouse. This pickups use only the pre-established routes given by one of the approaches, therefore it might be impossible to pick up all necessary products due to insufficient available vehicles' capacity. In order to reduce the distance traveled with loads destined to stock, contributing for less fuel consumption, the procedure attempts to pick up storable products from farmers which are visited closest to the end of the route. The implemented procedure does not order products by any specific criterion. They are simply ordered by their *id* reference. Note that different criteria can be established. For example one may choose to prioritize products which have the most quantities yet to be picked up during that week.

First it is necessary to introduce some new notation.

- $minQ$ - vehicle's minimum available capacity from farmer $j \in Fa^R$ to the end of the *Route* $R$.

Before running Algorithm 27 for a delivery day $d \in D_W$, $Q_{iW}$ (quantity to be available in week $W$ of product $i$) is updated as follows

$$Q_{iW} = Q_{iW} - \sum_{k \in Cl} \sum_{j \in Fa} e_{ijkd} - (pa_{id} - \sum_{k \in Cl} e_{i,AR,kd}).$$

**Algorithm 27** *PickUp*($S$), for a given day d

**for** $l = 1...|S|$ **do**
  $R \leftarrow S_l$
  **for all** $j \in Fa^R$ from last to first **do**
    **if** $j$ is not the warehouse **then**
      Let $x$ be the position of farmer $j$ in route $R$
      **if** $y^R_{x+1} < Q_v$ **then** ▷ verifies load arriving at the node following farmer $j$
        **for all** $i \in P_s$ **do**
          **if** $Q_{iW} > 0$ **then** ▷ if it is necessary to collect such product
            **if** $p_{ijd} > 0$ **then**
              $f_{ij,AR} = \min\{minQ, p_{ijd}, Q_{iW}\}$ ▷ quantity to be brought to the warehouse
              **if** $f_{ij,AR} \neq 0$ **then**
                $p_{ijd} = p_{ijd} - f_{ij,AR}$
                **for all** $s = (x+1)...|r^R|$ **do** ▷ locations visited after farmer $j$
                  $y^R_s \leftarrow y^R_s + f_{ij,AR}$ ▷ update loads
                **end for**
                $Q_{iW} = Q_{iW} - f_{ij,AR}$ ▷ updates weekly quantities to be picked up
              **end if**
            **end if**
          **end if**
        **end for**
      **end if**
    **end if**
  **end for**
**end for**
**return** $S$

For the example, and considering the solution obtained through the heuristic (Figure 5.7), we obtain the final distributional route (Figure 6.1).

Weekly quantities yet to pick up of product 2 = 1000 .
Max available capacity **135** ( 400 - 265).

**Final route**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Route** | [ depot | W | F_1 | F_2 | Cl_2 | Cl_3 | Cl_1 | depot ] |
| **Distance** | [ 0 | 0 | 0 | 1.5 | 11.5 | 17 | 26.5 | 29.5 ] |
| **Time** | [ 0 | 0 | 10 | 27 | 81 | 103 | 138 | 156 ] |
| **Load** | [ 0 | 0 | 20 | 370 | 400 | 394 | 273 | **135** ] |

Weekly quantities yet to pick up of product 2 = 1000 – 135 = 865.

Figure 6.1: Example - final solution.

## 6.2 Final Distribution Routes Planning

At the end of week $W$, after running Algorithm 27 for all delivery days, the quantity of product $i \in Ps$ ($W \in Tp^i$) that still has to be delivered to the warehouse is $Q_{iW}$. In the end of the week if there are still storable products to be brought to the warehouse, final pickup routes may be performed. The administrator has previously established a limit on the quantities of storable products to be picked up in the end of the week based on economic issues. The routes are only determined on the administrator's decision. This procedure is not currently implemented.

The determination of the final pickup routes is a capacitated *vehicle routing problem*(VRP) with time windows [16]. The vehicles leave empty from the depot, collect the storable products and return to the depot.

# Chapter 7

# Computational results

In this chapter, we present the computational results for the integrated approach and for the unaggregated approach. Seven instances, based on the case study, and the case study itself were tested. As for the former instances, different combinations of the number of customers and farmers were defined (Table 7.1). The number of fresh and storable products is always the same for these instances, three and fifteen, respectively.

In order to solve the presented MILP model, a free MILP solver software *lpsolve* 5.5.2.0 [17] was embedded in the developed software. This solver would be able to create the appropriate model and to solve it but, due to the excessive CPU time needed to solve even the smallest instance, it was established that the model would be constructed using lpsolve but it would be solved by the commercial CPLEX 12.5 [5] instead. Instances 1, 2, 3, 6 and 7 were solve to optimality, but in order to obtain a feasible solution for instances 4, 5 and the test case, a CPU time limit of one hour was established. The presented results for the MILP model were obtained using a desktop computer with an Intel Core 2 - 2.13 GHz processor and 2 GB RAM, and a Windows 7 Professional operating system.

As for the presented results regarding the unaggregated approach, this were obtained using a laptop computer with a Dual Core 2 - 2 GHz processor and 4 GB RAM, and a 32 bits Windows Vista operating system. The assignment of production to demand was made using the *fromFarmers1()* procedure instead of *fromFarmers2()*. As for the determination of the pickup and delivery routes, no intensive testing was made regarding the optimization of the simulated annealing parameters. The results presented are relative to the best solution obtained in ten turns using the following parameter setting:

- ACWSA : Parallel
- Overal method iterations : 15
- SA_1 : initial temperature = 500, final temperature = 1, Lundy & Mees, $\beta = 0.05$
- SA_2 : initial temperature = 500, final temperature = 1, Lundy & Mees, $\beta = 0.01$

Table 7.1: Instances.

| Instance | Customers | Suppliers |
|---|---|---|
| 1 | 3 | 3 |
| 2 | 4 | 3 |
| 3 | 5 | 3 |
| 4 | 6 | 3 |
| 5 | 7 | 3 |
| 6 | 3 | 4 |
| 7 | 3 | 5 |
| Study case | 7 | 5 |

Table 7.2: Solutions.

| Instance | Integrated approach | Time (s) | Unaggregated approach | Time (s) |
|---|---|---|---|---|
| 1 | 29.5 | 0.22 | 29.5 | 28 |
| 2 | 34.5 | 0.48 | 36 | 30 |
| 3 | 35 | 85.21 | 35.5 | 41 |
| 4 | 42 | 3600 | 48 | 66 |
| 5 | 45.8 | 3600 | 47.3 | 73 |
| 6 | 52 | 0.61 | 52 | 50 |
| 7 | 52.2 | 1.37 | 101.2 | 40 |
| Study Case | 110.2 | 3600 | 108.4 | 71 |

The results are summarized in Table 7.2.

The unaggregated approach addressed the computational complexity of the problem more successfully for the case study than the integrated approach, either in terms of costs or time consumption. The problem is solved to optimality in two instances with the former approach and in five instances with the integrated approach.

# Chapter 8

# Conclusions and Future Work

This work describes a Decision Support System that has been developed to help a set of farmers from the region of Setúbal to establish a short distribution channel, to deliver their production mainly to local canteens and restaurants. This DSS is composed of a data collection platform, a data storage module and software that establishes weekly distribution plans. Farmers can access all relevant information regarding themselves, customers, products, requests and production yields. The distribution planning problem encompasses the assignment of production to demand and the determination of the distribution routes. The assignment of production to demand takes into account that some products are storable.

Two different approaches where proposed and developed to optimize the daily distributional plan, an integrated approach and an unaggregated approach. These approaches assign the available production to customers' demand and establish pickup and delivery routes in an attempt to minimize the total transportation costs. As for the former approach, a mixed integer linear programming model was proposed. In the second approach, those problems are addressed separately and they are both solved using heuristics. Computational results showed that the use of exact methods was not suitable, since the inherent combinatorial complexity of the integrated problem made it impossible to obtain the optimal solution in a reasonable amount of time. The unaggregated approach revealed to be a good alternative to optimize the daily distributional plan. For the case study, it provided a better solution than the best solution found through the optimization of the MILP model, and it used much less computational time.

We can evaluate the unaggregated approach according to three main aspects: accuracy, efficiency and speed. Accuracy measures the degree of departure of the heuristic solution value from the optimal value, while efficiency regards the consistency of the heuristic, this is, if the approach is able to provide a "good" solution most of the times. Speed is also a concern since the combinatorial complexity of the problem may result in prohibitive computational times. Regarding accuracy we can state that it is not very good since it fails to achieve the optimal solution most of the times although usually is very near. As for efficiency we can state, even without thorough testing, that due to the random nature of the simulated annealing process, it is

not very efficient. The results vary a great deal. Regarding speed, it became obvious that, even without the proper simulated annealing parameters optimization, the unaggregated approach, was able to provide, for the case study, a better solution than the MILP model, in a relatively short period of time. The pickup and delivery heuristic may be improved by establishing different local search procedures or even develop different heuristics for the problem. Further improvement of the presented integrated MILP model is also advisable. New constraints may be added in order to strengthen the linear relaxation of the model and speed up the branch-and-bound.

The presented DSS is not fully developed. Regarding the collection of data, there are still issues to be addressed such as security and authentication procedures during the online submission of information. It is also necessary to provide the participants (customers and farmers) with the possibility of submitting their expected monthly demands and production yields. Regarding the software that establishes weekly distribution plans, it is still necessary to conclude the Weekly procedure, more specifically to implement the final distribution routes planning, in which routes to pickup storable products to the warehouse are determined.

Many other features can be introduced to improve the DSS like the calculation of the weekly profits and the inclusion of other information such as vehicle information. Distance calculations and route display will greatly benefit from the inclusion of an embedded *geographic information system* (GIS). This system would be used to easily calculate all distances and travel times between the participants and to provide a graphic display of the daily pickup and delivery routes. It is also necessary to provide farmers with the ability to modify distribution routes obtained by the DSS, displaying the respective changes in the overall solution (in terms of total distribution costs and quantities to be brought to the warehouse, for example).

# Bibliography


[1] Antunes. C.H., Gaspar-Cunha, A., Takahashi R., *Manual de Computação Evolutiva e Metaheuristica*, Coimbra University Press, pp. 163-167, 2012.

[2] Bosona, T., Gebresenbet, G., Ljungberg, D. and Nordmark, I., Integrated logistics network for the supply chain of locally produced food, Part I: Location and route optimization analyses, *Journal of Service Science and Management*, Volume 4, pp. 174-183, 2011.

[3] Bosona, T., Gebresenbet, G., Ljungberg, D. and Nordmark, I.,Integrated logistics network for the supply chain of locally produced food, Part II: Assessment of E-trade, economic benefit and environmental impact, *Journal of Service Science and Management*, Volume 5, pp. 249-262, 2012.

[4] Cortés, C.E., Matamala, M., Contardo, C., (2010) The pickup and delivery problem with transfers: Formulation and a branch-and-cut solution method, *European Journal Of Operational Research*, Volume 200(3), pp. 711-724.

[5] CPLEX 12.5, IBM Corp., 2013.

[6] Feenstra, G., Garret, S., Growing a community food system, *A Western Regional Extension Publication WREP0135*, 1999.

[7] López, J.F., A mixed integer programming model for a continuous move transportation problem with service constraints , *InnOvaciOnes de NegOciOs*, Volume 7(1), pp. 25-49, 2010.

[8] Mitrovic-Minic, S., Pickup and delivery problem with time windows: A survey. *Technical report TR 1998-12*, School of Computing Science, Simon Fraser University, Burnaby, BC, Canada,1998 .

[9] Parragh, S.N., Doerner, K.F., Hartl R.F., A survey on pickup and delivery problems Part I: Transportation between pickup and delivery locations, *JFB*, Volume 58, pp. 21-51, 2008.

[10] Parragh, S.N., Doerner, K.F., Hartl R.F., A survey on pickup and delivery problems Part II: Transportation between pickup and delivery locations, *JFB*, Volume 58, pp. 81-117, 2008.

[11] Ramakrishnan, R., Gehrke, J., *Database Management Systems* $3^{rd}$ edition, McGraw-Hill Education, 2011.

[12] Rand, G.K., The life and times of the Savings Algorithm for Vehicle Routing Problems, *Orion*, Volume 25(2), pp. 125-145, 2009.

[13] Rocha, A.A., *Estruturas de Dados e Algoritmos em Java*, FCA, 2011.

[14] Savelsberg, M.W.P., Sol, M., "The general pickup and delivery problem", *Transportation Science* 29(1), pp. 17-29, 1995.

[15] Taha H.A., Operations Research An Introduction., Prentice Hall, 2011

[16] Toth P., Vigo D., *The Vehicle Routing Problem*, 2002.

[17] www.lpsolve.sourceforge.net/5.5/

[18] www.php.net/manual/en/

# Appendix A

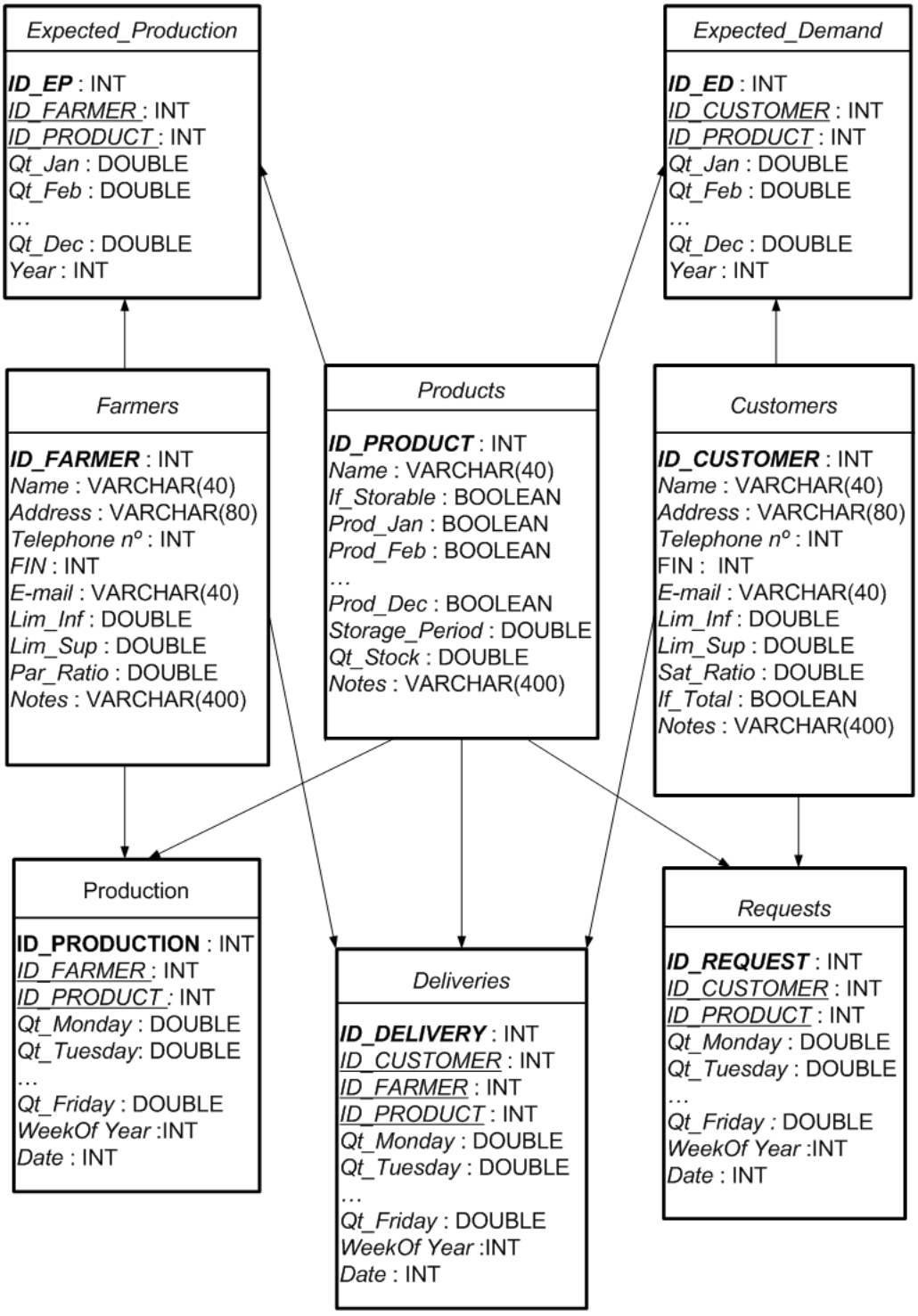



Figure 8.1: Database diagram